Instituto Superior de Engenharia do Porto

DESCOBERTA DE CONHECIMENTO EM LOGS DE TENTATIVAS DE INTRUSÃO

Um Estudo de Caso em Instituições de Ensino Superior

João Afonso Sarmento Morais

Dissertação para obtenção do Grau de Mestre em

Engenharia Informática

Área de Especialização em

Tecnologias do Conhecimento e Decisão

Orientador: Doutor Paulo Oliveira

Juri:

Presidente:

Doutora Maria de Fátima Coutinho Rodrigues

Vogais:

Doutor José Alberto da Silva Freitas

Doutor Paulo Jorge Machado Oliveira

Porto, Novembro 2010

Agradecimentos

Agradeço ao Professor Paulo Oliveira por ter viabilizado este trabalho e ter estado sempre disponível para o fazer avançar.

Quem também foi fundamental para o seu desenvolvimento, e a quem estou muito grato, foram os administradores de sistemas e redes que aceitaram colaborar comigo e permitir a instalação de um sistema de recolha de tentativas ilícitas de acesso, nas redes a seu cargo. Foram eles, por ordem alfabética, Eng. André David, Eng. André Moreira, Dr. Élio Coutinho e Eng. José Mendonça.

Por último, uma palavra de agradecimento ao amigo Ricardo Sousa, sempre pronto para ajudar, e aos meus pais, pois sem eles, não teria sido possível começar, nem acabar a tese.

Resumo

Perante a evolução constante da Internet, a sua utilização é quase obrigatória. Através da *web*, é possível conferir extractos bancários, fazer compras em países longínquos, pagar serviços sem sair de casa, entre muitos outros. Há inúmeras alternativas de utilização desta rede.

Ao se tornar tão útil e próxima das pessoas, estas começaram também a ganhar mais conhecimentos informáticos. Na Internet, estão também publicados vários guias para intrusão ilícita em sistemas, assim como manuais para outras práticas criminosas. Este tipo de informação, aliado à crescente capacidade informática do utilizador, teve como resultado uma alteração nos paradigmas de segurança informática actual.

Actualmente, em segurança informática a preocupação com o *hardware* é menor, sendo o principal objectivo a salvaguarda dos dados e continuidade dos serviços. Isto deve-se fundamentalmente à dependência das organizações nos seus dados digitais e, cada vez mais, dos serviços que disponibilizam *online*. Dada a mudança dos perigos e do que se pretende proteger, também os mecanismos de segurança devem ser alterados. Torna-se necessário conhecer o atacante, podendo prever o que o motiva e o que pretende atacar.

Neste contexto, propôs-se a implementação de sistemas de registo de tentativas de acesso ilícitas em cinco instituições de ensino superior e posterior análise da informação recolhida com auxílio de técnicas de *data mining* (mineração de dados). Esta solução é pouco utilizada com este intuito em investigação, pelo que foi necessário procurar analogias com outras áreas de aplicação para recolher documentação relevante para a sua implementação.

A solução resultante revelou-se eficaz, tendo levado ao desenvolvimento de uma aplicação de fusão de *logs* das aplicações Honeyd e Snort (responsável também pelo seu tratamento, preparação e disponibilização num ficheiro *Comma Separated Values* (CSV), acrescentando conhecimento sobre o que se pode obter estatisticamente e revelando características úteis e previamente desconhecidas dos atacantes. Este conhecimento pode ser utilizado por um administrador de sistemas para melhorar o desempenho dos seus mecanismos de segurança, tais como *firewalls* e *Intrusion Detection Systems* (IDS).

Palavras-chave: honeypot, honeyd, log, intrusão, mineração de dados, descoberta de conhecimento

Abstract

Internet's utilization is becoming more and more common. It's almost mandatory. Through the web it's possible to check bank statements, buy products from different countries, pay service bills, just to name a few.

In spite of this usability and closeness to people, Internet users started to have more computer science knowlegde. There are also manuals and guides giving detailed instructions about how to break in systems, among other criminal activities. All these facts, together with the growing user knowlege, changed today's systems and network security paradigms.

Nowadays, computer security in a company is closely related to the protection of data and service availability other than computer integrity like in the old days. This is fundamentally due to the growing dependency of organizations on their digital data and online services.

In order to follow the changing needs of security we must also change the security mecanisms. It's becoming imperative to know the intruder's motivation and goal.

With this in mind, the implementation of an intrusion logging system in five colleges was proposed, as well as its data analysis and interpretation using data mining techniques, although the combination of these concepts isn´t a common goal. In spite of this singularity, it was necessary to find analogue objectives in order to enable the gathering of relevant information to the implementation of the solution.

The achieved solution was considered to be effective as it was able to increase the previous knowledge obtained through statistical observation, revealing some of the attackers' useful characteristics. This new knowledge can be implemented by system administrators in their security mechanisms, like firewalls and Intrusion Detection Systems (IDS). An application that combines the logs of the software Honeyd and Snort was also developed. This new application is capable of cleaning and preparing information as well as making it available in a standard Comma Separated Values (CSV) format so it can be used by other applications.

Este conhecimento pode ser utilizado por um administrador de sistemas para melhorar o desempenho dos seus mecanismos de segurança, tais como *firewalls* e *Intrusion Detection Systems* (IDS).

Keywords: honeypot, honeyd, log, intrusion, data mining, knowledge discovery

Conteúdo

Agradecimentos **iii**

Resumo **v**

Abstract **vii**

1 Introdução **1**
1.1 Enquadramento 1
1.2 Objectivos 2
1.3 Tecnologias utilizadas 2
1.4 Organização da dissertação 3

2 Os novos perigos da Internet e uma solução de apoio à segurança **5**
2.1 Identificação do problema 5
2.2 Necessidade de soluções 6
2.3 Falhas das soluções mais adoptadas 7
2.4 Solução proposta 8

3 Solução de recolha de dados **9**
3.1 Sistemas de Registo de Acessos Ilícitos 9
3.2 *Honeypots* 10
3.2.1 *Honeypots* de baixa interacção 13
3.2.2 *Honeypots* de alta interacção 14
3.3 *Internet telescopes*, *Internet motion sensors* ou *darknets* 15
3.4 Sistemas de recolha e partilha de *logs* de *firewalls* 17
3.5 Definição da solução 19

4 Implementação da solução de recolha de dados **21**
4.1 Entidades colaboradoras e equipamento disponibilizado 21
4.2 Implementação do Honeyd 22
4.3 Implementação do Snort 26

4.4 *Scripts* para tolerância a falhas e gestão dos *logs* . . . 29
4.5 Desenvolvimento da aplicação de preparação dos dados . . . 32
4.6 Resultados obtidos . . . 34

5 Metodologias de mineração de dados em *logs* 37
5.1 *Data mining* . . . 37
5.2 *Logs* . . . 38
5.3 Mineração de dados em *logs* . . . 39
5.3.1 *Web mining* . . . 39
5.3.2 KDD . . . 40
5.3.3 Detecção de intrusões . . . 41
5.4 Definição das metodologias a implementar . . . 43

6 Mineração dos dados obtidos 45
6.1 Descrição do processo adoptado . . . 45
6.2 Resultados obtidos . . . 46
6.2.1 ISEP - servidor público (de 19/12/2009 a 18/05/2010) . . . 48
6.2.2 ICBAS - servidor externo (de 15/12/2009 a 17/05/2010) . . . 51
6.2.3 FADEUP - servidor externo (de 26/12/2009 a 18/05/2010) . . . 53
6.2.4 FCUP - servidor externo (de 11/12/2009 a 17/05/2010) . . . 55
6.2.5 FEP - servidor externo (de 18/01/2010 a 18/05/2010) . . . 57
6.2.6 Comparação . . . 59

7 Conclusão 63
7.1 Objectivos realizados . . . 63
7.2 Trabalho futuro . . . 64

Bibliografia 65

Glossário 71

Anexos 75

A Resultados obtidos com o SPSS Clementine 12.0 75

B Exemplo de configuração do Snort 89

C Excerto de ficheiro de *log* do Honeyd 109

D Excerto de ficheiro de *log* do Snort 111

E Excerto do ficheiro resultante da aplicação desenvolvida 115

Lista de Figuras

3.1 Esquema de implementação de honeynets 12
3.2 Esquema do funcionamento de uma *darknet* 16
3.3 Esquema do funcionamento da aplicação ThreatStop 18

4.1 Log do Snort no modo análise de pacotes (esq.), Honeyd (dir. em cima) e Snort no modo detecção de ataques (dir. em baixo) 27
4.2 Algoritmo da aplicação desenvolvida para preparação dos dados recolhidos 33

6.1 Exemplo de resultados obtidos com o algoritmo C5.0 47

Lista de Tabelas

3.1 Comparativo de *honeypots* de alta e baixa interacção 11

4.1 Campos que resultam da aplicação de tratamento de dados 34

6.1 Comparativo dos resultados obtidos para cada faculdade 60

Capítulo 1

Introdução

1.1 Enquadramento

A utilização da Internet é cada vez mais comum. Abrange serviços públicos cujo único modo de funcionamento é "através da Internet". Muitas empresas assentam já o seu modo de funcionamento na "*web*" sendo totalmente dependentes desta. O Hospital de Braga implementou este ano uma Plataforma de Sistemas de Informação *paper-free* (e foi o primeiro em Portugal), o que implica a paralisação total em caso de falha dos sistemas [Saúde, 2010].

Toda esta dependência tem alterado a forma como se encara a segurança informática. Actualmente, o principal objectivo desta é garantir a continuidade do funcionamento dos sistemas e a salvaguarda da informação. Os sistemas são agora parte fundamental no funcionamento das organizações.

Esta problemática é dificultada pelas conhecidas vulnerabilidades dos sistemas, a facilidade de troca de informação sobre metodologias de ataque e consequente quantidade e qualidade dos atacantes. Cada vez é mais fácil aprender a atacar sistemas e camuflar a sua origem.

Explorando vulnerabilidades de um sistema operativo, por exemplo, um atacante pode ganhar controlo sobre um servidor de uma instituição. A partir daí, consegue consultar informação confidencial, sabotar sistemas, impedir o funcionamento de objectos de negócio, entre muitas outras práticas criminosas, prejudicando a instituição atacada, podendo causar-lhe prejuízos avultados.

Perante o contínuo crescimento de ameaças informáticas, em número e complexidade, tornou-se fundamental conhecer as suas origens e motivações. Os sistemas tradicionais de protecção, tais como *firewalls* e sistemas de detecção de intrusão, não são capazes de garantir a ausência de tentativas de acesso ilícitas. Através de conhecimento prévio sobre os atacantes, a defesa dos sistemas e redes é favorecida no sentido em que pode actuar em antecipação ou até mesmo travar ataques em curso.

Para adquirir este tipo de informação é comum a implementação de sistemas de registo de acessos ilícitos. Estes analisam e registam tráfego não permitido ou que não deveria existir que pode posteriormente ser analisado. Existem principalmente três metodologias de registo. Honeypots que são sistemas ou recursos cuja a única função é serem acedidos ilicitamente e registar esses acessos; *dark-*

nets que resumidamente são sistemas de monitorização de tráfego dirigido a endereços IP que não estão a ser usados (analogia com escuro, *dark*) dentro de uma determinada gama; e recolha e partilha de *logs* de *firewalls* e *Intrusion Detection Systems* (IDS), oriundos de uma grande quantidade de fontes heterogéneas.

1.2 Objectivos

Perante esta evolução das ameaças, e necessidade de melhoramento dos sistemas de segurança, os objectivos inicialmente propostos para este trabalho foram os seguintes:

- selecção do sistema de registo de acessos ilícitos mais adequado à extracção de conhecimento útil a administradores de sistemas e redes, com base no levantamento efectuado.
- implementação do sistema de registo de acessos ilícitos seleccionado em diferentes instituições e durante vários meses.
- desenvolvimento de uma ferramenta para o tratamento dos dados obtidos, introdução de outras informações importantes referentes a cada registo e a sua fusão, dado que provêm de duas aplicações diferentes e o seu formato é distinto. Pretende-se que a aplicação faça ainda a normalização da informação para o formato *Comma Separated Values* (CSV) com vista a poder ser utilizada correctamente por um programa de mineração de dados.
- selecção dos algoritmos mais apropriados para descoberta de conhecimento que será útil na configuração de sistemas de segurança.
- extracção de conhecimento através da utilização dos algoritmos anteriormente seleccionados.

1.3 Tecnologias utilizadas

A ferramenta escolhida para registo de acessos ilícitos foi o Honeyd por ser a mais utilizada em investigação, muito poderosa e de código aberto. A esta juntou-se o programa Snort em modo de detecção de ataques, de forma a acrescentar ao *log* de acessos o nome do ataque lançado. O Snort foi escolhido por ser também muito usado em investigação, habitualmente no modo de detecção de pacotes, com o intuito de recolher dados que permitam a futura criação de assinaturas de ataques.

Esta solução de recolha foi colocada em instituições que aceitaram participar, e foi mantida em funcionamento durante vários meses. Conseguiu-se a colaboração de cinco faculdades portuguesas. Estas disponibilizaram-se para permitir a instalação da solução de registo de acessos ilícitos criada nas suas redes. A solução baseou-se em *honeypots* de baixa interacção por terem um baixo risco de comprometimento, salvaguardando as redes das instituições colaboradoras. Por outro lado, emulam serviços Windows e Linux pelo que evitam a necessidade de aquisição de licenças.

Desenvolveu-se uma aplicação que integra os dados obtidos pelo Honeyd com os do Snort, normalizando-os e preparando-os para aumentar a quantidade de conhecimento interessante que se pode obter. Esta faz ainda a normalização da informação para o formato CSV.

A selecção dos algoritmos de extracção de conhecimento foi baseada em analogias com diferentes tipos de abordagens, dado que esta não é vulgar em investigação, pois não é comum registar acessos ilícitos com o objectivo de caracterizar o atacante e os seus ataques. Os trabalhos efectuados na área têm-se focado mais na identificação de novos ataques.

Para esta análise, encontrou-se similaridades com *web mining*, detecção de ataques e descoberta de conhecimento em bases de dados, e conclui-se que as metodologias mais utilizadas e apropriadas são algoritmos Apriori, K-means, Redes Neuronais e Análise Temporal.

A extracção de conhecimento foi efectuada com o *software* SPSS Clementine 12.0, a partir dos dados previamente tratados com a aplicação desenvolvida.

A abordagem adoptada e a conjugação das tecnologias utilizadas mostraram contribuir para um melhor conhecimento das ameaças e, consequentemente, para uma melhoria da segurança informática das organizações que foram caso de estudo.

1.4 Organização da dissertação

No capítulo 2 é feita a descrição do problema, assim como da sua importância e solução proposta. No capítulo 3 é apresentado um estudo sobre técnicas de registo de acessos ilícitos. Após esta exposição, segue-se a escolha da técnica que melhor se adequa à solução proposta, entre as anteriormente referidas.

A descrição da implementação da solução é feita no capítulo 4. Esta consiste na instalação de um *honeypot* de baixa-interação (Honeyd) em conjunto com um IDS (Snort), e no desenvolvimento de uma nova ferramenta para fusão e tratamento dos *logs* destas duas aplicações.

O capítulo 5 é dedicado a técnicas de mineração de dados, focando a sua aplicabilidade em informação de *log*, e a sua utilidade na extracção de conhecimento sobre atacantes informáticos e as suas práticas criminosas. São também evidenciadas neste capítulo as técnicas consideradas mais apropriadas.

A aplicação das metodologias de mineração de dados seleccionadas, assim como a análise de resultados obtidos, é explanada no capítulo 6.

Por fim, no capítulo 7, começa-se por fazer um pequeno resumo do trabalho efectuado e da sua contextualização, seguindo-se a exposição das conclusões finais e propostas para trabalho futuro.

Capítulo 2

Os novos perigos da Internet e uma solução de apoio à segurança

2.1 Identificação do problema

Na ultima década a utilização da Internet ganhou carácter quase obrigatório, existindo já um grande número de serviços apenas acessíveis através desta rede. Deste então, o número de utilizadores cresceu para quase cinco vezes mais, sendo agora cerca de 1.734.000.000 [Stats, 2009]. Pode-se depreender que a utilização de computadores aumenta cada vez mais, assim como o conhecimento em informática por parte da população.

Este aumento provocou modificações na forma como a sociedade interage, simplificando muitos processos. As pessoas podem usar a Internet para comunicar, fazer compras ou aceder à sua conta bancária. Por outro lado surgiram novas formas de cometer crimes fomentadas pelo fácil acesso à Internet e pelo baixo risco para os criminosos [Grégio et al., 2007].

A fomentação da utilização de sistemas informáticos trouxe a disponibilização de informação e aplicações para ataques informáticos, assim como o aumento da capacidade de processamento a preços acessíveis a todos. Levou também ao aparecimento de mais *hackers*, *crackers* ou *coders*, indivíduos que tentam penetrar em sistemas informáticos de forma ilícita [Carsten, 2007].

As empresas têm alargado o uso de computadores nas suas indústrias. Isto leva a que sejam alvos fáceis para atacantes e corram riscos de perda. Por exemplo, se o sítio web que disponibiliza uma loja *online* bloqueia, vai implicar uma perda significativa de lucro. Para travar os atacantes têm-se usado *firewalls* mas estas não conseguem suster a totalidade dos ataques. Por isso têm sido desenvolvidos muitos sistemas de detecção (*(N)IDS - (Network) Intrusion Detection Systems*) e prevenção de intrusões ((N)IPS - *(Network) Intrusion Prevention Systems*) [Su, 2009]. Estes serão abordados mais à frente.

Antigamente, o objectivo da segurança informática era garantir o bom funcionamento do hardware pois esta era a parte mais cara dos *data centers*. Actualmente, este já não é tão caro e pode até ser alugado a baixo custo. O objectivo passou a ser garantir a segurança da informação e garantir a

continuidade de serviço pois as empresas assentam nestes pontos o seu funcionamento [Baldissera and Nunes, 2007].

2.2 Necessidade de soluções

É frequente ouvir-se falar da existência de vírus, *adware*, *spyware*, *trojans*, *fishing*, entre outros [Computer-Security.Org.UK, 2007]. Mais preocupantes para um administrador de sistemas são as vulnerabilidades dos sistemas operativos e aplicações. Por definição, "ataques ou ameaças informáticas são todas as acções que pressupõem uma violação da segurança do sistema, confidencialidade, integridade ou disponibilidade" [Carvalho, 2009] e é contra estas ocorrências que se torna cada vez mais difícil proteger os sistemas de uma instituição, como aplicações web e bases de dados, por exemplo [Rist, 2009].

Os anti-vírus reduziram bastante a propagação de infecções a partir de suportes físicos de dados. No entanto, as comunicações abriram outras vias de ataque para os *blackhats*. A sondagem massiva de serviços de rede e o impacto dos *worms* de rede tem sido controlada com o avanço tecnológico das *firewalls* e IDS. Mas o aumento de largura de banda expõe cada vez mais computadores às ameaças dos *blackhats*. Isto levou a que os *worms* sejam novamente vulgares. Apenas recentemente se conseguiram melhorias de segurança, principalmente nos sistemas Microsoft Windows, que reduziram a propagação deste tipo de ameaças. Estas melhorias fizeram com que os atacantes se focassem mais a nível aplicacional. Sistemas ou aplicações *online* são habitualmente os seus alvos, sendo o desvio de dinheiro o seu principal objectivo [Watson, 2007].

Quando se coloca esta problemática a instituições de ensino superior depara-se com outras questões, como a privacidade de dados confidenciais (contactos, moradas, salários), alteração de notas, comunicações a nível de voz e dados, acesso indevido a exames ou acesso a serviços (impressão, biblioteca, inscrições a exames, fotocópias, parque automóvel) de forma ilícita ou fazendo-se passar por outrem. Uma faculdade pode ter milhares de alunos o que implica um potencial número elevado de indivíduos interessados em violar os seus sistemas informáticos e colocar em risco o seu funcionamento académico e administrativo.

Consequentemente, a administração de servidores de uma instituição de ensino superior requer sistemas de segurança para tentar impedir acessos ilícitos, tais como encriptação, Sistemas de Detecção de Intrusão, Sistemas de Prevenção de Intrusão, filtros de pacotes a nível de conteúdo, portas, endereços IP e estado das ligações [Carvalho, 2009]. Pode-se ainda dividir a rede em Redes Locais Virtuais (VLAN - *Virtual Local Area Networks*), usar controlo de acessos por filtro de endereço mac, arp seguro, autenticação e autorização. Será suficiente?

Rui Reis no âmbito do seu projecto de Bacharelato do curso de Engenharia Informática do ISEP criou dois *honeypots* de alta interacção[1] ligados à Internet, um pela Netcabo outro pela SapoADSL e concluiu que a máquina ligada à Netcabo foi atacada um número de vezes que quase duplica o da outra

[1] Sistemas ou recursos informáticos cuja finalidade é serem comprometidos por atacantes e registar as todas as suas operações.

[Carvalho, 2009]. Após a análise dos ataques, concluiu também que 99% destes registos resultam de exploração de vulnerabilidades.

As organizações governamentais enfrentam diariamente atacantes cujo intuito é comprometer informação do governo, expor inseguranças nacionais e prejudicar a estabilidade financeira do país. As ameaças são reais [Hess, 2009].

Analisando este jogo de ataque/defesa percebe-se que o atacante tem sempre consigo a vantagem de ser ele que tem a iniciativa, ou seja, é ele que controla o confronto enquanto a linha de defesa tenta impedir a sua progressão [Project, 2006]. Mas as organizações podem usar sistemas de registo de acessos ilícitos para aprender mais sobre quem os tem como alvo e porquê [Project, 2006]. Que ferramentas usa, procedimentos, frequência, altura dos ataque, etc..

2.3 Falhas das soluções mais adoptadas

Sun Tzu, o antigo general Chinês, concluiu um dos princípios básicos da actualidade da segurança *online* dois milénios antes da electricidade ser inventada: "se conhecer o inimigo e a si próprio, não precisa temer o resultado de cem batalhas". Muitas organizações da actualidade gastam avultadas quantias em tecnologia defensiva. No entanto, poucas pararam para pensar e compreender a natureza das ameaças a que estão sujeitos. Quantas saberão quem os ataca e qual a sua motivação [Watson, 2007]?

A NSA (National Security Agency), empresa que garante a segurança digital nacional nos Estados Unidos da América, está a desenvolver um sistema de monitorização de indícios de ataques informáticos contra entidades públicas e privadas tais como companhias de electricidade ou estações de tratamento de água. Contudo têm surgido muitas questões relativamente a violação de privacidade e aos motivos da NSA [Bradley, 2010].

Os sistemas de protecção mais eficazes têm preços avultados. Outros sistemas apresentam grande dificuldade em distinguir com precisão acessos ou conteúdo legítimos de ilícitos, nomeadamente os IDS que são conhecidos pelo elevado número de casos incorrectamente classificados (falsos positivos)[O'Neill, 2007]. Contudo, os IDS e IPS podem ser afinados por forma a serem mais eficientes, detectando maior quantidade de tráfego ilícito sem prejudicar a troca de pacotes não comprometida.

Os IDS são sistemas que tentam detectar ataques assim que eles ocorrem ou depois de o ataque ter sido iniciado. O seu modo de funcionamento baseia-se em recolher e analisar tráfego em determinado ponto da rede ou sistema informático e depois usar esta informação para proteger a rede. Estes sistemas podem funcionar no modo de *misuse* (abuso) ou *anomaly* (anomalia). No modo de detecção de abuso apenas conseguem identificar ataques já conhecidos através da comparação do tráfego recolhido com assinaturas de ataques previamente identificados. Pelo contrário, no modo de detecção de anomalias conseguem detectar novos ataques por meio de utilização de métodos heurísticos que identificam algo anómalo no tráfego em relação ao que foi capturado anteriormente [Aydin et al., 2009].

Os sistemas de registo de acessos ilícitos vieram suplantar as limitações dos tradicionais IDS [Yeldi and et al., 2003]. Podem ser usados para recolher informações que identificam o atacante e responder

a questões sobre como se defender e derrotar intrusos cuja identidade é desconhecida, assim como o seu modo de operação e os seus motivos [Raynal et al., 2004].

Embora tenham a vantagem de poder analisar todo o tráfego que passa num dado ponto, não estando limitados a um determinado conjunto de endereços IP [Hoepers et al., 2007], a qualidade dos dados recolhidos pelo IDS não é boa para análise futura. Além do grande volume de dados, a maioria dos alertas são desprezáveis e levam ao desleixo da sua análise por parte do administrador. Com alguns sistemas de registo de acessos ilícitos tal não acontece porque todos os acessos são de alguma forma proibidos ou não deveriam existir e, por isso, consideram-se tentativas de ataque [Gaudin, 2002].

2.4 Solução proposta

A recolha de dados relacionados com ameaças da Internet tornou-se uma tarefa de segurança habitual para os investigadores de segurança ou administradores de redes. No entanto, devido à grande quantidade e diversidade dos dados pode ser muito difícil analisá-los [Thonnard and Dacier, 2008].

Como solução propõe-se a utilização de técnicas de *data mining* para combater a dificuldade de análise dos dados recolhidos. Esta abordagem tem por objectivo a determinação de padrões sobre os atacantes de instituições de ensino superior, nomeadamente quem são e quais as suas motivações. Este tipo de análise não tem sido muito explorado em investigação, tendo-se procurado mais identificar novos ataques do que os atacantes e o seu modo de operação.

Pretende-se implementar um sistema que englobe a fase de recolha e preparação de dados em várias instituições de ensino superior. Para a preparação de dados será desenvolvida uma aplicação que, muito resumidamente, os tratará filtrando-os, derivando novas informações que possam ser interessantes e uniformizando o resultado final num ficheiro CSV. Estes sistemas deverão funcionar durante alguns meses e, posteriormente, a informação que recolheram será analisada através de técnicas de *data mining* e interpretada por forma tentar identificar conhecimento útil a um administrador de sistemas/rede nas suas tarefas de prevenção e reacção a tentativas de ataque.

Nos capítulos seguintes serão abordadas técnicas de registo de acessos ilícitos e *data mining* e a forma que se optou para os conjugar. Será também descrita a implementação desta solução em cinco instituições de ensino superior que aceitaram colaborar. Além desta, refere-se em detalhe o desenvolvimento da aplicação de preparação de dados. Por fim, será efectuada a demonstração e interpretação de resultados obtidos através da aplicação das técnicas de *data mining* escolhidas.

Capítulo 3

Solução de recolha de dados

3.1 Sistemas de Registo de Acessos Ilícitos

Tal como é descrito no capítulo anterior, a solução proposta passa por registar acessos ilícitos durante alguns meses em instituições que aceitem colaborar. Para este efeito, existem diversos sistemas que se enquadram numas de três abordagens possíveis.

A primeira consiste na utilização de *honeypots*, sistemas cuja implementação apenas tem como objectivo o registo de todos os acessos que lhes são feitos, não tendo qualquer função verdadeiramente útil para os utilizadores da rede [Riordan et al., 2006; Provos, 2004; Baecher et al., 2006].

A segunda abordagem baseia-se nos *Internet telescopes*, *Internet motion sensors* ou *darknets*. Estas designações são sinónimos para sistemas que permitem que uma máquina observadora consiga monitorizar eventos de larga escala a decorrer na Internet. São usados para monitorizar todo o tráfego dirigido a endereços IP que não estão a ser usados (analogia com escuro, *dark*) dentro de uma determinada gama [Moore et al., 2004; Cymru, 2010]. Dado que estes endereços não são usados, o tráfego que lhes é dirigido é suspeito, podendo corresponder a ataques ou a má configuração de algum sistema.

A última classe de técnicas de recolha de informação sobre ataques a sistemas consiste na recolha e partilha de *logs* de *firewalls* e *Intrusion Detections Systems* (IDS), oriundos de uma grande quantidade de fontes heterogéneas. Uma implementação desta abordagem é o sistema DShield que recebe *logs* do mundo inteiro, provenientes de instituições voluntárias. Além disto, disponibiliza a informação correlacionada ao público [Thonnard and Dacier, 2008].

Segue-se uma descrição pormenorizada destes conceitos e os seus mais recentes desenvolvimentos. No final deste capítulo é descrita a solução a implementar e os factores que levaram à sua escolha.

3.2 *Honeypots*

Os *Honeypots* têm vindo a ganhar popularidade à medida que os profissionais de segurança informática têm efectuado estudos mais detalhados sobre o estado da arte das práticas dos atacantes. Além destas, a sua motivação, hábitos e padrões comportamentais [Gaudin, 2002].

Estes sistemas são recursos cujo o grande valor reside no seu uso ilícito ou não autorizado. Esta é a definição desenvolvida por Lance Spitzner fundador do projecto *The Honeynet Project* e líder não oficial da comunidade *honeypot*. Mais à frente será descrito o que são *honeynets*. Voltando à definição anterior, ela inclui os dois princípios básicos: um *honeypot* pode ser qualquer sistema computorizado, desde impressoras, a *routers*, a uma rede inteira; é propositadamente configurado de forma a permitir o seu comprometimento e não oferece nenhum recurso verdadeiramente útil ao utilizador, servindo apenas para registar os acessos que lhe vão sendo efectuados [Spitzner, 2003d].

A sua verdadeira função é ser sondado, atacado ou comprometido para que as acções dos intrusos possam sem observadas, analisadas e compreendidas [Spitzner, 2003a]. Não têm nenhuma interacção autorizada pelo que qualquer interacção é considerada ilícita ou maliciosa. O conceito é extremamente simples mas também extremamente eficaz [Spitzner, 2003a].

Há muitas definições que vão sendo atribuídas e conhecidas tais como: solução para ludibriar atacantes; tecnologia usada para detectar ataques; computadores configurados para serem atacados e se aprender com esses acessos. Todas estas definições estão correctas [Spitzner, 2003e]. Devem ser usados como um complemento para a segurança da rede de uma instituição e não devem ser encarados como substitutos para boas práticas de segurança, políticas de segurança, sistemas de gestão de correcções de segurança (*patches*) e outras ferramentas de segurança, como *firewalls* e IDS [Hoepers et al., 2007].

Se um servidor de email for constantemente atacado, a análise dos seus *logs* não fazem dele um *honeypot*. Apenas pode ser considerado um servidor mal configurado a nível de segurança [Spitzner, 2003d]. Um *honeypot* não tem que ser um sistema computorizado. Pode ser qualquer recurso com o qual se pretende que um atacante interaja. Estes recursos que não são computorizados são conhecidos por *honeytokens*. São recursos digitais cujo valor é apenas ser usado inadvertidamente, à semelhança dos *honeypots*. Por exemplo, um número de cartão de crédito, um documento, um registo ou um *login* falso [Spitzner, 2003d].

Quando um atacante percebe que está perante um *honeypot* fica habitualmente embaraçado e zangado. A sua reacção será provavelmente tentar um ataque destrutivo na rede real [Gaudin, 2002]. Poderá também tentar falsear os dados da análise através de geração propositada de tráfego. Por isso, a instituição deve ponderar bem qual o grau de risco que quer correr [Project, 2006].

Outra desvantagem é o facto de poderem ser comprometidos e, nessa situação, lançar ataques a partir do interior da rede [Sardana and Joshi, 2009]. Por isto devem ser criadas redes específicas apenas para implementação de *honeypots* e sem contacto com as outras máquinas da rede. O atacante pode ainda tirar o sistema de funcionamento ou parar a recolha de informação. Embora possa não executar qualquer operação nociva aos sistemas da rede, convém ter em conta que pode usá-la para

Tabela 3.1: Comparativo de *honeypots* de alta e baixa interacção

Características	**Honeypot de baixa interactividade**	**Honeypot de alta interactividade / Honeynet**
Instalação	fácil	mais difícil
Manutenção	fácil	trabalhosa
Risco de comprometimento	baixo	alto
Obtenção de informações	limitada	extensiva
Necessidade de mecanismos de contenção	não	sim
Atacante tem acesso ao S.O. real	não	sim
Aplicações e serviços oferecidos	emulados	reais
Atacante pode comprometer o *honeypot*	não (em teoria)	sim

actividade criminal, tal como contrabando de filmes, música, cartões de crédito roubados ou pornografia infantil [Project, 2006].

Podem ser implementados de duas formas: produção ou baixa interacção; e pesquisa ou alta interacção. A tabela 3.1 ajuda a esclarecer estes conceitos, tendo sido retirada de [Hoepers et al., 2007].

No caso da produção, os sistemas ou recursos não são reais. São emulado por *scripts* ou aplicações que simulam serviços dando a ideia ao visitante de que está perante servidores autênticos. Daí a baixa interacção, visto que esta está limitada a um conjunto fixo de situações previstas na emulação. Servem principalmente para melhorar a segurança de uma organização, ajudando a detectar intrusões [Veerasamy et al., 2009]. Oferecem baixo risco de comprometimento e por isso são ideais para instituições que não podem correr o risco de implementar um *honeypot* de alta interacção [Hoepers et al., 2007].

A implementação de pesquisa é feita recorrendo a recursos reais. Por isso se chamam habitualmente de alta interacção, dado que a interacção é total. Têm por objectivos explorar as operações do atacante, identificar falhas e vulnerabilidades e melhorar a segurança do sistema [Veerasamy et al., 2009]. Esta forma de implementação aumenta o risco da rede pois os sistemas estão muito mais expostos, e em caso de comprometimento o atacante tem um vasto leque de opções de ataque ou acesso ilícito [Hoepers et al., 2007].

Um outro tipo específico de *honeypots* são as *honeynets*. Estas consistem em *honeypots* de alta interacção, desenhados para recolherem grandes quantidades de informação relativa a ameaças, internas ou externas à rede. O que as torna diferentes é o facto de formarem uma rede de computadores e outros recursos reais para interagir com atacantes [Project, 2006]. Por isso devem conter mecanismos de controlo para impedir que sejam usadas contra outras redes, além dos referidos sistemas de captura

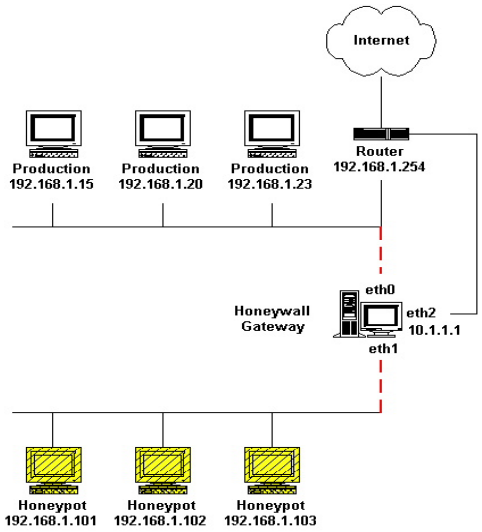

Figura 3.1: Esquema de implementação de honeynets

de dados e geração de alertas.

A figura 3.1 exemplifica uma possível implementação, tendo sido retirada de [Project, 2006]. Nesta é também ilustrado o conceito de *honeywall*, recurso além de ser o *gateway* da *honeynet* é também responsável pelo controlo e recolha de tráfego. Funciona normalmente no nível dois da camada OSI[1] e é a fronteira entre a rede de *honeypots* e a rede de produção. Por isso, é um dos elementos mais críticos das *honeynets*.

As *honeynets* dividem-se em dois tipos, as reais e as virtuais. A única diferença entre elas é o facto de os sistemas serem implementados sobre máquinas virtuais ou reais/físicas. As *honeynets* reais têm a vantagem de enganarem melhor o atacante e serem mais tolerantes a falhas por estarem distribuídas por várias máquinas. Em contrapartida, as virtuais são habitualmente criadas usando uma única máquina contendo uma plataforma de virtualização, tal como VMWare, XEN ou Hyper-V. Isto representa um ponto único de falha. No entanto, torna-se mais barata de implementar, mais fácil de gerir e não requer tanto espaço físico para os equipamentos [Hoepers et al., 2007]. Permite ainda gestão gráfica remota, nomeadamente a criação ou recriação de sistemas à distância. O backup é facilitado com a possibilidade de guardar e restaurar posteriormente servidores inteiros, e o consumo de energia é inferior [Watson, 2007].

As *honeynets* virtuais dividem-se ainda em auto-contenção e híbridas. As de auto-contenção implementam todos os mecanismos, incluindo contenção, captura de dados, geração de alertas e os *honeypots*, num único computador. As híbridas separam os mecanismos de contenção, captura de dados e geração de alertas, por dispositivos distintos, e apenas os *honeypots* são implementados num único computador através de *software* de virtualização [Hoepers et al., 2007].

[1]O nível ou camada dois corresponde à ligação de dados, onde já são suportadas Virtual Local Area Networks (VLAN) mas não endereços IP.

Nos sub-capítulos seguintes apresentam-se alternativas de implementação de *honeypots*. Embora algumas das aplicações já não disponham de actualizações, são todas referidas por terem sempre diferenças no modo de operação e funcionalidade, que podem influenciar a escolha de uma ferramenta.

3.2.1 *Honeypots* de baixa interacção

Desde 2003 a ferramenta de monitorização de acessos ilícitos mais usada em investigação tem sido o Honeyd que permite criar *honeypots* de baixa interacção. A sua aplicabilidade tem sido demonstrada em diversos países, quer em ambientes académicos, quer industriais. Criado por Niels Provos em 2002, escrito em C e desenhado para plataformas Linux, introduziu uma variedade de novos conceitos, nomeadamente a capacidade de monitorizar milhares de IPs não utilizados e simular muitos sistemas operativos ao mesmo tempo [Spitzner, 2003b]. Tem também a capacidade de monitorizar acessos TCP, UDP e ICMP. Existe uma versão para Windows de código aberto chamada WinHoneyd desenvolvida pela netVigilance [netVigilance, 2010]. Estes *honeypots* têm-se mostrado fáceis de identificar devido ao seu conjunto limitado de respostas. Contudo, a sua popularidade é grande para efeitos de recolha de informação pelas vantagens descritas anteriormente [Thonnard and Dacier, 2008].

Seguem-se outros exemplos de desenvolvimentos na área [Spitzner, 2003b; SourceForge, 2010; netVigilance, 2010; Fielding, 2007]:

- Bubblegum Proxypot - um *proxy honeypot* para ludibriar e detectar *spammers*.
- Jackpot - Um *relay honeypot* aberto também destinado a *spammers*.
- BackOfficer Friendly - BOF - aplicação gratuita para Windows cuja finalidade é alertar quando são detectados intrusos. Criado por Marcus Ranum e a equipa NFR em 1998, BOF é muito fácil de usar e corre em qualquer plataforma Windows. Contudo é muito limitado e só consegue escutar em 7 portas.
- Bait-n-Switch - não é um *honeypot* mas uma tecnologia que permite direccionar todo o tráfego não autorizado ou destinado a zonas não utilizadas para *honeypots*.
- Bigeye - *honeypot* que emula diversos serviços.
- HoneyWeb - emula diferentes tipos de servidores *web*. Tem a capacidade de se adaptar dinamicamente consoante o tipo de pedido que lhe é efectuado.
- Deception Toolkit - DTK - o primeiro *honeypot* de código aberto, lançado em 1997. Escrito por Fred Cohen, DTK é uma colecção de scripts Perl e código fonte C que emula diversos serviços. O seu principal propósito é enganar atacantes humanos. Esta ferramenta está desactualizada, e apenas é mencionada por ser uma referência no segmento.
- LaBrea Tarpit - *honeypot* de código aberto que difere dos outros por ter sido desenvolvido para atrasar ou parar ataques, agindo como um *sticky honeypot*. É compatível com Windows e Linux.

- Sendmail SPAM Trap - *honeypot* que identifica *spammers* e captura o seu SPAM sem o enviar para as suas vítimas.
- Tiny Honeypot - criado por George Bakos, a sua grande diferença é parecer sempre vulnerável ao atacante. Qualquer que seja o ataque lançado irá sempre parecer bem sucedido.
- Valhala Honeypot - *honeypot* de código aberto desenvolvido para Windows que permite emular servidores web, ftp, finger, telnet, smtp, pop3, tftp e possibilita o envio de *logs* remotamente.
- Nepenthes - desenvolvido em 2006 permite emular vulnerabilidades conhecidas e capturar worms à medida que o tentam infectar. Não permite identificar novas explorações de vulnerabilidades.

3.2.2 *Honeypots* de alta interacção

O uso de *honeynets* tem sido dificultado por questões legais. Para alguns sistemas jurídicos a sua utilização é uma violação de privacidade por serem capazes de monitorizar todos os movimentos de quem lhes acede. O seu custo de implementação elevado e dificuldade de gestão devido à sua distribuição tem levado à utilização de tecnologias de virtualização [Thonnard and Dacier, 2008].

Apesar destas dificuldades, têm sido usadas com sucesso na detecção de ataques contra sistemas operativos comuns e na determinação do tempo necessário ao seu comprometimento, mantendo as suas configurações por omissão.

Seguem-se alguns exemplos reconhecidos da utilidade da implementação de *honeypots* de alta interacção [Watson, 2007]:

- *reverse engineering* até ao dia zero de *malware*, *worms* e ferramentas de detecção em massa/*auto-rooters*.
- publicação de um dos primeiros casos documentados de troca secreta de cartões de crédito e crimes informáticos organizados.
- detecção contínua de criminosos informáticos envolvidos em esquemas financeiros, tais como *carding*.
- captura e categorização de *phishing* e *pharming*, em conjunto com SPAM, operações de *proxy* públicos, *theft* e Distributed Denial of Service (DDOS) *extortion rackets*.
- observação de atacantes a configurar os seus sites públicos, *forums*, servidores IRC em *honeypots* comprometidos.
- participação em movimentações políticas em diversos países, tais como Índia, Paquistão ou Indonésia.
- criação de um dos primeiros modelos de classificação de perfis de sociabilização de *blackhats* através da captura de dados de IRC e observação do mundo real.

- detecção de tráfego invulgar, tal como tráfego IPv6 sobre IPv4 originado por um grupo italiano de *hackers* que tentavam penetrar nos sistemas da NASA via servidores Solaris situados no México.
- identificação de sessões de *hacking* mapeadas sobre protocolos de voz sobre rede e ligações ponto-a-ponto maliciosas em redes de produção.
- análise da propagação de *malware* e *botnets* e consequente criação de um modelo de detecção e controlo e soluções para o seu impedimento.
- identificação de *botnets* e outros mecanismos de sondagem a ser usados na Internet para recolha de informação de cartões de crédito e credenciais.

3.3 *Internet telescopes*, *Internet motion sensors* ou *darknets*

Moore et al propuseram as *darknets* como uma alternativa aos existentes sistemas de medida e monitorização de redes [Gagadis and Wolthusen, 2008]. Utiliza-se sempre nos capítulos seguintes o termo *darknet* e não expressões semelhantes (como *Internet telescopes* ou *Internet motion sensors*) para evitar que a utilização de diferentes nomes dificulte a compreensão dos conceitos.

Este tipo de sistemas tem vindo a ser cada vez mais utilizado por administradores na monitorização de tráfego anómalo vindo do exterior. São uma técnica de identificação de tráfego inesperado ou não solicitado. Têm por princípio básico a monitorização de tráfego destinado a gamas de endereços IP válidas, mas não utilizadas, isto é, que não estão associadas a nenhuma actividade legítima. Logo, não haverá tráfego de resposta a estes pedidos. São, por isso, considerados sistemas de recolha passivos [Harrop and Armitage, 2005].

O tráfego recolhido pode corresponder a má configuração de dispositivos de rede, ataques de negação de serviço em massa, *malware*, ou varrimento de portas e endereços IP para geração de um mapa da rede, tal como é demonstrado na figura 3.2 [Bailey et al., 2005; Gagadis and Wolthusen, 2008]. A origem deste tipo de tráfego não se apercebe, habitualmente, que está a enviar pacotes para um espaço monitorizado. São também conhecidos por “Black Holes” (buracos negros) ou “Internet Sinks” (ralos da Internet) [Cooke et al., 2004].

Têm a grande vantagem de poder complementar sistemas de detecção de vírus e de tentativas de intrusão que se prevê que irão suceder ou que estão a acontecer no momento. São muito fáceis de implementar e há muitas soluções gratuitas. Conseguem detectar tráfego ilícito sem a necessidade de complexas análises de assinatura, assumindo que nenhum pacote deveria ser dirigido a uma *darknet*.

As *darknets* de empresas e *campus* universitários, internas e pequenas, trazem grandes vantagens a estas instituições, nomeadamente, a rápida detecção de máquinas internas a farejar outras partes da rede. As ameaças que se encontram dentro da rede são da maior importância dado que o seu acesso a outros sistemas da rede não é, habitualmente, controlado por mecanismos de defesa. Ao contrário das implementações para investigação, estas não têm por objectivo a detecção de actividade alargada na Internet, excepto no caso de serem ameaçadas directamente por este tipo de actividade [Harrop and Armitage, 2005; Gagadis and Wolthusen, 2008].

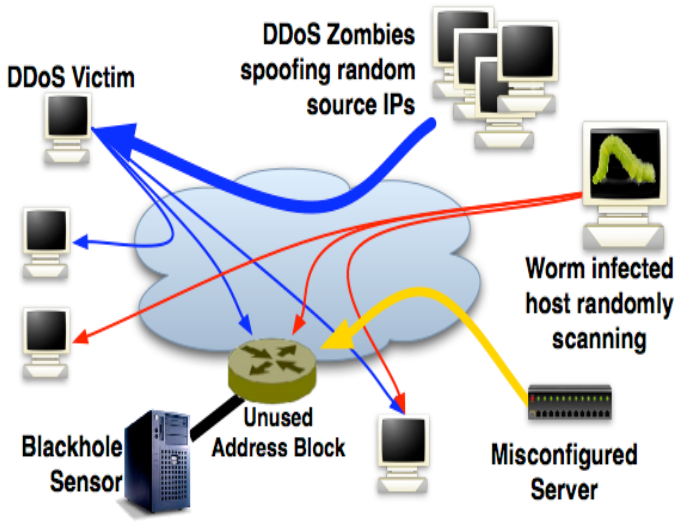

Figura 3.2: Esquema do funcionamento de uma *darknet*

Outros trabalhos de investigação têm-se focado na implementação de pequenas *darknets* e na sua utilidade numa LAN. Estes trabalhos têm como finalidade criar métodos eficazes de permitir identificar *worms* o mais rapidamente possível, tendo por base padrões de escuta [Moore et al., 2004; Bailey and Cooke, 2004; Cooke et al., 2004].

Contudo, têm-se vindo a desenvolver soluções que requerem uma grande quantidade de blocos de IPs contínuos, apresentando como mínimos redes com máscara de vinte e quatro bits, dezasseis ou até menos (redes maiores). Isto não se revela um problema para ISPs ou redes de investigação, contudo pode ser impeditivo para muitas empresas e pequenos ISPs [Harrop and Armitage, 2005].

Existem várias formas de implementação de *darknets*. As *darknets* distribuídas, por exemplo, consistem numa junção de várias pequenas *darknets* numa só grande *darknet*. Esta abordagem permite abranger muito mais gamas de endereços [Gagadis and Wolthusen, 2008].

Um dos maiores projectos de *darknets* é o *Internet Motion Sensor* (IMS) que utiliza uma distribuição de *darknets*, grandes e pequenas, pelo espaço Internet. As redes mais pequenas utilizadas têm máscara de vinte e quatro bits [Harrop and Armitage, 2005].

A capacidade do IMS baseia-se numa *darknet* distribuída pelo mundo, por dezoito organizações, empresas e redes académicas, em três continentes. Monitoriza aproximadamente dezassete milhões de endereços e, passados dois anos e meio de ter entrado em funcionamento, recebia uma média de nove pacotes por segundo [Bailey et al., 2006]. Tem a capacidade de diferenciar e categorizar tráfego. Suporta ainda análise de dados em tempo real. Por outro lado, devido à grande quantidade de dados, no caso de falhas durante o seu processamento, o sistema poderá reportar uma grande quantidade de falsos positivos ou negativos. Para melhorar a eficiência do processamento, este é feito pelos sensores e não numa base de dados central [Gagadis and Wolthusen, 2008].

Outra abordagem possível são *darknets anycast*. Para as implementar, criam-se rotas em vários

pontos da rede que apontam para a mesma rede local, e apenas esta rede local é monitorizada. Esta abordagem não permite observar tantos endereços como as *darknets* distribuídas. Por isto, habitualmente, a quantidade de dados recolhida é pequena. No entanto, possibilita um processamento mais rápido da informação devido à sua concentração numa única gama de endereços [Moore et al., 2004].

A terceira variação de *darknets* são as *transit network*. Neste caso, a monitorização abrange várias redes, à semelhança das distribuídas, mas distingue-se por apenas analisar tráfego no interior das redes e não nas suas extremidades. Esta diferença traz vantagens relativamente à distribuição e sincronização de dados. Contudo, tem a desvantagem de apenas monitorizar tráfego que efectivamente entra na rede, tendo uma abrangência bastante mais reduzida que as *darknets* distribuídas [Harrop and Armitage, 2005; Moore et al., 2004].

Existem também as *honeyfarm darknets*. Nesta abordagem, a monitorização não é efectuada passivamente como nas outras. Os sistemas, além da monitorização, interagem com as máquinas remotas através de *honeypots*. Isto traz muitas implicações pois as *darknets* podem monitorizar muitos endereços em simultâneo. Assim, necessitam de decidir quais os endereços que irão responder activamente a tráfego detectado. Têm também que definir a que tráfego é que vão responder devido à possibilidade de ocorrência de vários eventos em simultâneo e não serem capazes de responder a tudo. O carácter activo destas *darknets* tem ainda a desvantagem de aumentar muito a ocupação da rede, podendo interferir quer com a monitorização, quer com a utilização normal da rede [Harrop and Armitage, 2005; Moore et al., 2004].

Por último, referem-se as *greynets*. Estas diferem das *darknets* por usarem endereços IPs válidos, juntamente com os não válidos. Desta forma, aumentam a probabilidade de detecção de fenómenos anormais na rede, tais como ataques de *malware* [Gagadis and Wolthusen, 2008; Harrop and Armitage, 2005].

3.4 Sistemas de recolha e partilha de *logs* de *firewalls*

Os sistemas de recolha e partilha de *logs* de *firewalls* têm sido desenvolvidos por organizações com o intuito de modificar o panorama da protecção informática. Pretendem auxiliar os sistemas de segurança existentes a impedir a ocorrência de ataques conhecidos, assim como de novos ataques. Seguem-se alguns dos projectos existentes.

O projecto mais conhecido é o Internet Storm Center (ISC) que recolhe, correlaciona e analisa informação recolhida por mais de três mil *firewalls* e *IDS*. A recolha é feita por sensores compatíveis com a maioria do *hardware* e sistemas operativos, denominados Storm Center Analsys and Coordination Centers (SACCs). Estes SACCs estão dispostos por várias organizações, ISPs e entidades independentes. Esta solução cobre mais de quinhentos milhões de endereços IP em mais de cinquenta países. Permite identificar quais os serviços mais atacados, os IPs considerados mais perigosos e indicadores de novos *worms* ou explorações de vulnerabilidades [Center, 2010; Brugger, 2004; Searchers, 2010].

Os SACs enviam a informação que recolhem para uma base de dados comum chamada DShield. Esta suporta vários tipos de *logs* de *firewalls*, IDS e outros equipamentos de rede. O ISC disponibiliza

Figura 3.3: Esquema do funcionamento da aplicação ThreatStop

ao público a informação recolhida. Permite ainda questionar a sua base de dados relativamente à existência de registos com determinados endereços IP de origem e destino [Brugger, 2004; Center, 2010].

Existem outras aplicações semelhantes tais como o myNetWatchman. Outro projecto equivalente é o DeepSight Analyzer cuja grande diferença para os anteriores é o cariz comercial, pois não disponibiliza a sua informação de forma gratuita [Brugger, 2004].

Uma outra abordagem é a da Shadowserver Foundation, que consiste num grupo de profissionais de segurança voluntários, e tem por objectivo recolher, identificar e reportar *malware*, *botnets* e fraude electrónica. Apesar de não ter fins lucrativos, trabalha com outras agências de segurança no sentido de desenvolver estratégias e planos de acção para fazer frente às ameaças à medida que vão surgindo [Shadowserver, 2010].

O PhishTank é também um projecto colaborativo de recolha de dados de segurança, particularmente sobre *fishing* na Internet. Proporciona também, gratuitamente, uma API para que programadores e investigadores possam incluir informação *anti-fishing* nas suas aplicações [PhishTank, 2010].

Outra iniciativa relevante é o projecto de pesquisa Cyber-Threat Analytics (Cyber-TA). Tem por objectivo apoiar as organizações a defenderem-se contra ameaças da Internet. Para isto, pretende desenvolver centros de análise de ameaças digitais. Estes serão completamente automatizados e escaláveis para suportar as crescentes quantidades de dados. Terão também a capacidade de reconhecimento de ataques, prioritizando-os e isolando os mais críticos. O projecto conta com investigadores notáveis em intrusões em redes de larga escala e privacidade da informação e ambiciona desenvolver tecnologias colaborativas inovadoras [Cyber-TA, 2010].

Por ultimo, refere-se o ThreatStop. Este projecto comercial usa milhares de sensores distribuídos pela Internet para identificar criminosos informáticos. Consiste numa aplicação que impede que os com-

putadores de uma rede comuniquem com estas entidades, através do bloqueio de tráfego vindo de uma determinada lista de endereços IP, denominada ThreatList. A ThreatList é actualizada frequentemente, recorrendo aos projectos ISC, Cyber-TA, Shadowserver e PhishTank. As *firewalls* conseguem actualizar as suas regras de acordo com esta lista, através de DNS. A figura 3.3 esquematiza o funcionamento da aplicação [Inc., 2010].

3.5 Definição da solução

Para a solução a implementar optou-se por usar *honeypots* devido à sua interactividade com o atacante e viabilidade da cooperação com instituições. Embora no caso de utilização de *darknets* também se possa considerar que todo o tráfego é ilícito, têm a desvantagem de ser menos ricas em dados, comparativamente com os *honeypots* dado que um acesso a um IP (não utilizado) não obterá resposta e por isso não haverá interactividade. Pelo contrário, na utilização de recolha e partilha de *logs* de *firewall* e IDS, a informação consiste em acessos a IPs de servidores existentes, contudo, mesmo após selecção da parte relevante dos *logs*, estes são menos detalhados e dificilmente se conseguiria o mesmo grau de independência da solução por se tratar com *firewalls* e IDS que são sistemas críticos das instituições.

A escolha do tipo de *honeypot* recaiu sobre os de baixa interacção. Considerou-se preferível o risco de serem identificados mais facilmente do que o de comprometerem a segurança da rede. Os *honeypots* de alta interacção ou *honeynets* foram considerados demasiado perigosos e dispendiosos, o que dificultaria bastante a colaboração de instituições. Não só requeriam mais recursos como obrigavam à aquisição de licenças para os cenários Windows.

O Honeyd foi a ferramenta escolhida principalmente por ser actualmente a mais utilizada em investigação. Isto faz com que haja muita informação em torno da melhor forma de utilização e problemas conhecidos. Acresce o facto de ser poderosa a nível de funcionalidades, versátil a nível de capacidade de emulação e escuta e de código livre [Mikhalenko, 2006]. Da pesquisa efectuada notou-se que o programa Snort é muito usado para captura de tráfego ao nível do pacote. Tem também a particularidade de poder funcionar em modo de detecção de ataques em que regista o nome do ataque que detectou, caso o identifique. Com base nesta funcionalidade, optou-se por implementar o Snort juntamente com o Honeyd, por forma a adicionar à informação recolhida pelo Honeyd o nome de alguns ataques detectados. Desta forma, a solução implementada foi composta por estes dois programas.

Para a solução de recolha pretendia-se definir um bloco de endereçamento IP da instituição, que fosse roteável e que não estivesse a ser utilizado. Este bloco deveria ser visto como uma rede isolada ou um novo segmento de rede da instituição, não pertencendo a qualquer rede ou segmento de rede que já estivesse a ser utilizado. A poluição de dados como, por exemplo, o acesso para gestão do *honeypot* via rede ou verificação da disponibilidade dos serviços, foi planeada para poder ser facilmente removida dos *logs*. Caso contrário, podia ser extremamente difícil distinguir entre os eventos que faziam parte dos procedimentos de administração e manutenção e os eventos gerados por actividades maliciosas [Hoepers et al., 2007].

Habitualmente usam-se diferentes sistemas operativos que respondem num conjunto de portas conhecidas. Os sistemas podem ser implementados por forma a serem acedidos apenas por utilizadores internos da rede ou expostos ao público, para uma percepção de como variam as ameaças consoante a sua exposição. Devem ser inseridos no DNS para corroborar o disfarce, respondendo também aos pedidos de DNS invertidos [Thonnard and Dacier, 2008].

Convidaram-se cinco faculdades a participar, e todas aceitaram a instalação da solução descrita na sua rede. Esta foi mantida em funcionamento durante vários meses, obrigando a uma constante monitorização e salvaguarda dos dados. A informação recolhida foi depois processada com uma ferramenta desenvolvida propositadamente para o efeito, pois além do tratamento habitual de dados houve a necessidade de fundir o *log* proveniente do Honeyd com o do Snort. A descrição detalhada desta implementação é efectuada no capítulo seguinte.

Capítulo 4

Implementação da solução de recolha de dados

4.1 Entidades colaboradoras e equipamento disponibilizado

A análise de informação resultante de tentativas de ataque necessita de dados fidedignos e actuais. Para tal, solicitou-se a colaboração de cinco instituições de ensino superior no sentido de disponibilizarem a sua rede e máquinas para a implementação de um sistema de recolha de dados de tentativas de ataque. Nomeadamente, o Instituto de Ciências Biomédicas Abel Salazar (ICBAS) da Universidade do Porto (UP), Faculdade de Ciências da UP (FCUP), Faculdade de Desporto da UP (FADEUP), Faculdade de Economia da UP (FEP) e Instituto de Superior de Engenharia do Porto (ISEP) do Instituto Politécnico do Porto (IPP).

A solução de recolha de informação escolhida foi, tal como é descrito no capítulo anterior, a conjugação entre o *software* de implementação de *honeypots* Honeyd e o sistema de detecção e prevenção de intrusões Snort. O Honeyd pode simular diversos serviços em sistemas operativos diferentes e registar todas as tentativas de acesso a estes. O Snort tem sido usado em investigação principalmente no modo recolha de tráfego ao nível do pacote, para caracterização de ataques. Dado que se pretendia caracterizar o atacante, o Snort foi utilizado no modo de detecção de ataques. Neste modo regista apenas acessos suspeitos e a um nível mais alto que inclui o tipo de ataque identificado.

Algumas das faculdades colaboradoras acrescentaram aos seus serviços de DNS os nomes dos *honeypots*. Estes foram definidos por forma a não se desmascararem, do género “hd1” e não “honeypot1”. Foram criadas regras nas *firewalls* de modo a que os sistemas simulados não tivessem restrições de acesso ao exterior e fossem acessíveis da rede interna ou pública conforme se tratasse de um *honeypot* interno ou externo. Em todas as faculdade colaboradoras foi cedido um acesso remoto à(s) máquina(s) para gestão. Houve o cuidado de efectuar este acesso sempre através do mesmo endereço IP. As operações de gestão do *honeypot* via rede ou verificação da disponibilidade dos serviços são incompatíveis com a análise pretendida e, desta forma, facilitou-se a sua eliminação.

Nas referidas instituições de ensino já são usadas plataformas de virtualização centralizadas à excepção da FADEUP, pelo que se optou por criar máquinas virtuais para instalação da solução de recolha, com 256MB de RAM e 8GB de disco. Esta opção é justificada com a isenção de custos, dado que apenas requer recursos de memória, disco e processamento disponíveis. Para além disto, não coloca em risco o hardware tal como aconteceria numa instalação física. Tem ainda vantagens a nível de espaço físico e gestão remota. Na FADEUP foi disponibilizada uma máquina física PIV com 512MB de RAM e 80GB de disco.

As entidades ICBAS, FCUP e FEP usam o XenServer. Como havia recursos suficientes foram criadas duas máquinas virtuais em cada uma das instituições, uma que simulava servidores internos e outra externos. Contudo, o XenServer apresenta uma limitação de base na criação de máquinas virtuais, só permitindo que seja instalado um número limitado de sistemas operativos (SO). Nenhum destes SO contém pacotes de instalação pré-compilados da ultima versão do Honeyd (1.5c). Optou-se então pela instalação do Linux CentOS5.5 e compilação manual do software.

4.2 Implementação do Honeyd

O serviço Honeyd foi configurado para emular 4 servidores, *email* em Windows, *email* em Linux, *web* em Windows e *web* em Linux, mais concretamente em Microsoft Windows 2000 Server SP3 e Linux 2.3.28-33. Estes serviços foram escolhidos por serem os mais utilizados nas faculdades e, como tal, os de maior importância. Como exemplo, apresenta-se um ficheiro de configuração usado:

```
create default
 set default default tcp action block
 set default default udp action block
 set default default icmp action block

create win-mail
 set win-mail personality "Microsoft Windows 2000 Server SP3"
 add win-mail tcp port 25 "sh /usr/local/honeyd/share/honeyd/scripts/win/exchange-smtp.sh"
 add win-mail tcp port 110 "sh /usr/local/honeyd/share/honeyd/scripts/win/exchange-pop3.sh"
 add win-mail tcp port 143 open
 add win-mail tcp port 465 open
 add lnx-mail tcp port 587 open
 add win-mail tcp port 993 open
 add win-mail tcp port 995 open
 set win-mail default tcp action reset
 set win-mail default udp action reset
 set win-mail default icmp action open
 set win-mail uptime 3423653

create win-web
```

```
 set win-web personality "Microsoft Windows 2000 Server SP3"
 add win-web tcp port 80 "sh /usr/local/honeyd/share/honeyd/scripts/win/iis.sh"
 add win-web tcp port 443 open
 set win-web default tcp action reset
 set win-web default udp action reset
 set win-web default icmp action open
 set win-web uptime 1564446

create lnx-mail
 set lnx-mail personality "Linux 2.3.28-33"
 add lnx-mail tcp port 25 "perl /usr/local/honeyd/share/honeyd/scripts/lnx/smtp.pl"
 add lnx-mail tcp port 110 "sh /usr/local/honeyd/share/honeyd/scripts/lnx/emulate-pop3.sh
  $ipsrc $sport"
 add lnx-mail tcp port 143 open
 add lnx-mail tcp port 465 open
 add lnx-mail tcp port 587 open
 add lnx-mail tcp port 993 open
 add lnx-mail tcp port 995 open
 set lnx-mail default tcp action reset
 set lnx-mail default udp action reset
 set lnx-mail default icmp action open
 set lnx-mail uid 24521 gid 14521
 set lnx-mail uptime 1332336

create lnx-web
 set lnx-web personality "Linux 2.3.28-33"
 add lnx-web tcp port 80 "/usr/bin/python
  /usr/local/honeyd/share/honeyd/scripts/lnx/HoneyWeb-0.4/HoneyWeb-0.4.py
    'Apache/1.3.27 (Unix) mod_perl/1.27' $ipsrc $sport $ipdst"
 add lnx-web tcp port 443 open
 set lnx-web default tcp action reset
 set lnx-web default udp action reset
 set lnx-web default icmp action open
 set lnx-web uid 15361 gid 15361
 set lnx-web uptime 2136324

bind 192.168.1.82 win-mail
bind 192.168.1.83 win-web
bind 192.168.1.84 lnx-mail
bind 192.168.1.85 lnx-web
```

Os sistemas operativos usados na simulação foram escolhidos com base no facto de as plataformas Windows e Linux serem as mais usadas para implementação de servidores. Pretendeu-se dar a possibilidade de relacionar os ataques com o sistema operativo dos servidores, após análise futura dos *logs*. As versões de sistema operativo usadas são antigas para que o atacante assuma que o sistema está desactualizado e não tenha dificuldade em encontrar informação sobre as suas vulnerabilidades conhecidas. Por outro lado, o ficheiro nmap.prints contido no Honeyd não contem assinaturas de sistemas operativos recentes. Optou-se por não substituir estas assinaturas com o intuito de não correr o

risco de utilizar um ficheiro não conforme, mantendo a compatibilidade com o *nmap* e *xprobe*[1].

A configuração da emulação do servidor de *email* aceita conexões nas portas 25 (SMTP), 110 (POP3), 143 (IMAP), 465(SMTPS), 587 (SMTPS), 993 (IMAPS) e 995 (POP3S) por serem usadas habitualmente pelos servidores de email e para aumentar a probabilidade de ataque, embora só responda nas portas 25 (SMTP) e 110 (POP3). Apenas estas portas são usadas para resposta porque são as usadas habitualmente para os serviços emulados (POP3 e SMTP). Para a emulação do servidor web optou-se por aceitar conexões nas portas 80 (HTTP) e 443 (HTTPS), embora só responda na porta 80 por ser a que é usada habitualmente para o serviço emulado (HTTP). Foram abertas as mesmas portas para a situação Windows e Linux por uma questão de igualdade de exposição. Com a intenção de aumento da exposição foi permitida a entrada de pedidos ICMP.

O servidor *web* emulado apenas responde na porta 80 através de um *script*, embora também aceite conexões na porta 443. O parâmetro "reset" fecha uma determinada porta enquanto que "open" a abre. Se usados na definição "default tcp/udp/icmp action" têm a mesma finalidade mas para todas as portas cujo comportamento da emulação não foi definido. O parâmetro "block" é usado para rejeitar conexões. O perfil "default" foi criado apenas por precaução e só é usado quando as conexões não fazem correspondência com as instruções "bind". Estas são as que definem a correspondência entre os endereços IP dos servidores emulados e os seus perfis de emulação.

Embora o Honeyd permita que o servidor emulado responda a múltiplos endereços IP de um segmento de rede, optou-se por usar apenas um IP para cada servidor simulado por uma questão de proximidade com o mundo real e por limitações a nível de IPs disponíveis em alguma instituições.

Houve o cuidado de alterar os valores "uptime", "uid" e "gid" para que não fossem iguais aos valores introduzidos por omissão, facilitando a identificação do sistema como um *honeypot*, conforme as boas práticas da implementação Honeyd referidas no seu manual. Também foram alterados os *scripts* de simulação de serviços pelo mesmo motivo, através da introdução de conteúdo em português referentes à instituição em que foram implementados. Após as alterações foram devidamente testados. O Honeyd traz alguns por omissão e podem ser encontrados outros no sítio oficial. Surgiram dificuldades de implementação dos *scripts* devido à falta do pacote gcc. Outros apresentavam mesmo incoerências no código, pelo que foram todos revistos. Nomeadamente, a definição de *timeouts* para impedir o bloqueio dos servidores por ataques de *Denial of Service* (DoS), que foi muito frequente inicialmente, até a alteração ser efectuada.

Todos os acessos efectuados aos servidores emulados pelo Honeyd são registados. O formato deste registo não pode ser alterado. Há apenas a possibilidade de ser canalizado para o *log* do sistema (*syslog*). Optou-se por utilizar para este efeito um ficheiro à parte. O programa é lançado com o comando:

[1] Ferramentas que permitem identificar o sistema operativo de uma máquina remota.

```
/usr/local/honeyd/bin/honeyd -p /usr/local/honeyd/share/honeyd/nmap.prints
  -x /usr/local/honeyd/share/honeyd/xprobe2.conf -a
  /usr/local/honeyd/share/honeyd/nmap.assoc -f
  /usr/local/honeyd/share/honeyd/honeyd.conf -l
    /var/log/honeyd/honeydlog 192.168.1.82-192.168.1.85 --disable-webserver
```

As opções são:

- p - caminho para o ficheiro nmap.prints
- x - caminho para o ficheiro xprobe
- a - caminho para o ficheiro nmap.assoc, que associa as assinaturas nmap às assinaturas xprobe
- f - caminho para o ficheiro de configuração
- l - caminho para o ficheiro de *log*; caso não seja utilizado este parâmetro as mensagens vão para o syslog
- 192.168.1.82-192.168.1.85 - gama de endereços onde serão emulados os serviços; qualquer tentativa de acesso a um destes IPs fica registada
- disable-webserver - desactiva o servidor *web* interno do Honeyd pois não é necessário e desta forma liberta-se recursos

Para que o Honeyd responda aos pedidos efectuados aos IPs simulados, a máquina em que está implementado tem de responder a múltiplos IPs. Esta funcionalidade pode ser conseguida através do programa arpd. Este permite à máquina onde é executado receber tráfego dirigido aos IPs indicados por parâmetro, caso a configuração de rede o permita. Foi instalado através do comando:

```
rpm -ivh arpd* --force
```

O parâmetro “force” é necessário caso a instalação do arpd anuncie um conflito com outras bibliotecas. A utilização deste parâmetro não prejudica o funcionamento normal do sistema operativo. Após conclusão da instalação foi criado um *link* para que o programa encontrasse a biblioteca libdnet.1:

```
ln /usr/lib/libdnet.so.1 /usr/lib/libdnet.1
```

O programa é chamado através do comando "/usr/sbin/arpd IP_simulado", onde IP_simulado deve ser substituído pelo IP do *honeypot*.

No caso do ISEP é usada a tecnologia de virtualização VMWare. Esta não apresenta as limitações do XEN a nível de sistemas operativos das máquinas virtuais. Desta forma, optou-se por instalar a distribuição Fedora12, por já conter as ultimas versões do Honeyd e Snort nos seus repositórios. Para instalar este software basta usar o gestor de pacotes (yum, por exemplo).

Apenas foi disponibilizada uma máquina virtual mas com dois interfaces de rede físicos e 8 virtuais. Desta forma não foi necessário recorrer ao serviço arpd, pois configurou-se os IPs simulados nos interfaces virtuais. A execução dos programas Honeyd e Snort foi feita com o parâmetro "-i" onde se especificou qual o interface de rede físico a usar para aquela instância. Mais à frente no documento existe um exemplo desta execução, na parte do automatismo do arranque do sistema.

Foi necessária alteração a nível de routing devido à existência de dois interfaces de rede físicos, por forma a que a resposta ao tráfego oriundo do exterior fosse encaminhada para o interface exterior e vice-versa. Segue-se a alteração efectuada para a máquina do ISEP, onde o default gateway era o interface eth0:

```
route add -net 192.168.62.0 netmask 255.255.255.0 gw 172.31.0.1 eth1
route add -net 10.0.0.0 netmask 255.252.0.0 gw 172.31.0.1 eth1
route add -net 172.28.0.0 netmask 255.252.0.0 gw 172.31.0.1 eth1
route add -net 172.16.0.0 netmask 255.248.0.0 gw 172.31.0.1 eth1
```

A implementação da solução na FADEUP foi feita num computador físico, sem virtualização, não estando também limitada a nível de sistema operativo. Desta forma, seguiu o modelo de implementação do ISEP por uma questão de simplicidade da instalação do *software*, e a nível da configuração de rede por uma questão de coerência, dado que também se trata apenas de uma máquina a emular servidores internos e públicos.

4.3 Implementação do Snort

Estudou-se a hipótese de guardar o tráfego destas máquina a nível do pacote. É demonstrado um exemplo do funcionamento desta implementação do Snort em conjunto com o Honeyd na figura 4.1.

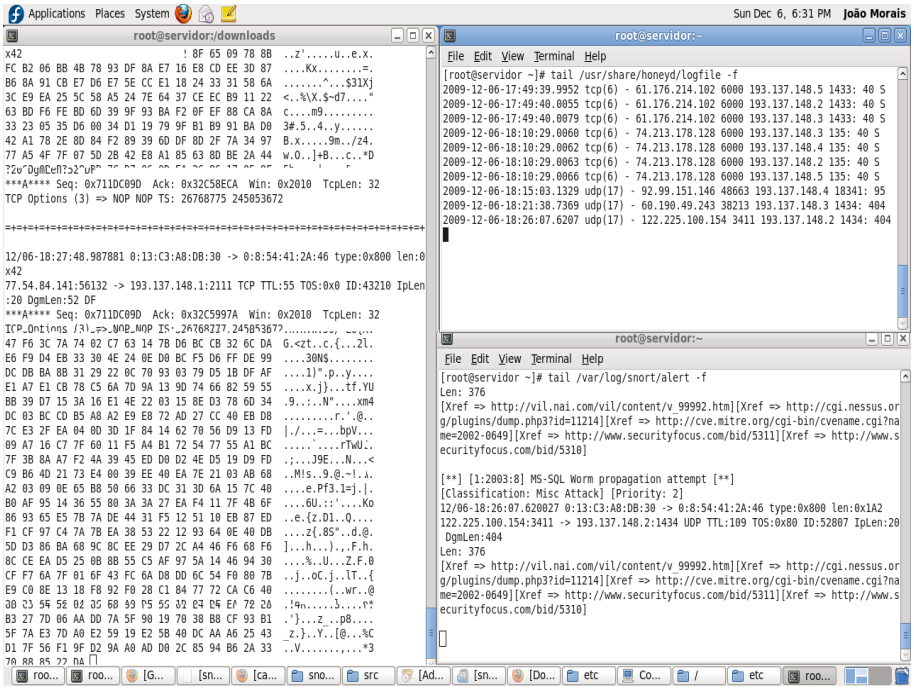

Figura 4.1: Log do Snort no modo análise de pacotes (esq.), Honeyd (dir. em cima) e Snort no modo detecção de ataques (dir. em baixo)

Esta opção podia permitir a definição de novas regras para IDS. Para o efeito, colocou-se o Snort em execução com o seguinte comando:

```
snort -l /root/downloads/testeSNORT/ -b &
```

Neste modo, o Snort detecta todo o tipo de tráfego que passa pela porta física da máquina onde está a correr. Correu-se a aplicação durante 2 semanas. Após o tratamento dos seus *logs* com filtros do programa Wireshark e uma pequena aplicação desenvolvida em Ruby, conseguiu-se obter a informação num formato normalizado. Contudo, verificou-se que todos os pacotes detectados eram diferentes entre si. Embora os dados não correspondessem a um período muito extenso de captura, a diversidade dos valores registados deixava antever o insucesso da sua mineração. Por outro lado, no caso do ISEP, esta solução gerava cerca de 15MB de *log* por dia. Isto causava problemas quer quanto ao espaço físico disponível, quer quanto à transferência dos resultados que iam sendo obtidos pela Internet, dado que a máquina só era acedida remotamente. Considerou-se que o rápido crescimento do volume de dados colocava em risco o bom funcionamento dos sistemas e prejudicava a sua manutenção. Dado

que esta abordagem divergia um pouco do tema inicial, o tempo disponível para conclusão do trabalho e o perigo de instabilidade do sistema desenvolvido, optou-se por usar o Snort apenas em modo de detecção de ataques. A descrição da configuração que se segue corresponde a este modo.

A instalação do Snort consiste em descarregar no site oficial o pacote de instalação específico para a versão de sistema operativo desejada e instalá-lo. Depois é necessário o download do conjunto de regras actualizado, o qual obriga a registo no site oficial. Estas regras devem ser descompactadas para o directório "/etc/snort/rules".

Para configurar o Snort usa-se o ficheiro "/etc/snort/snort.conf". Neste, deve-se comentar a linha que contém a opção "dcerpc2" porque impede o arranque do programa. Embora esta linha sirva para activar a verificação de tráfego Server Message Block (SMB), esta deve arrancar normalmente mesmo com o comentário referido anteriormente, apresentando a mensagem "DCE/RPC enabled" no início do arranque do programa. De qualquer forma, é prescindível para o trabalho a detecção e descodificação de tráfego SMB. Ao correr o programa, usando por exemplo o comando "snort -v", este deve acusar erro em linhas referentes a regras para as quais não são encontrados os respectivos ficheiros. Estas linhas devem ser comentadas, pois correspondem a regras que já não são actualizadas e disponibilizadas no site oficial.

O Snort foi configurado por forma a detectar o máximo número de evidências, uma vez que ao ser corrido num honeypot todo o tráfego detectado corresponde a tentativas de intrusão e não há o perigo de falsos positivos. A configuração usada é descrita no anexo B. A forma de arranque do programa, escolhida de acordo com a necessidade do trabalho foi:

```
/usr/sbin/snort -de -c /etc/snort/snort.conf --alert-before-pass --treat-drop-as-alert -D
```

As opções são:

- d - modo de escuta
- e - analisar os cabeçalhos dos pacotes
- c - caminho para o ficheiro de configuração
- alert-before-pass - alterar as regras de detecção, começando pelas regras de alerta antes das regras de passagem
- treat-drop-as-alert - considerar o tráfego seleccionado para ser bloqueado como alerta, passando desta forma a ser registado nos logs
- D - modo "deamon", para correr em segundo plano

Os *logs* são guardados no directório "/var/log/snort". Para indicação de um caminho diferente usa-se o parâmetro -l com indicação do directório desejado. Também se pode ignorar tráfego vindo de determinados IPs, o que é útil para ignorar máquinas que implementam serviços de monitorização, como o NAGIOS ou SpiderWorks, através da opção -F seguida do caminho para um ficheiro com o seguinte formato ("src" corresponde a endereço IP de origem e "dst" de destino):

```
not (src host 193.136.26.11) and
not (dst host 193.136.26.11)
```

4.4 *Scripts* para tolerância a falhas e gestão dos *logs*

Os servidores foram configurados para enviar os *logs* por *email* diariamente, para um melhor controlo do funcionamento dos sistemas, colocando no directório cron.daily o seguinte script:

```
(date ; uuencode /var/log/honeyd/honeyd-ext.log isep-hdexterno.txt) | mail -s
  ''Ficheiro honey ISEP ext actualizado'' 1000261@isep.ipp.pt
(date ; uuencode /var/log/snort-ext/alert isep-snortexterno.txt) | mail -s
  ''Ficheiro snort ISEP ext actualizado'' 1000261@isep.ipp.pt
(date ; uuencode /var/log/honeyd/honeyd-int.log isep-hdinterno.txt) | mail -s
  ''Ficheiro honey isep int actualizado'' 1000261@isep.ipp.pt
(date ; uuencode /var/log/snort-int/alert isep-snortinterno.txt) | mail -s
  ''Ficheiro snort isep int actualizado'' 1000261@isep.ipp.pt
```

Para o script anterior funcionar é necessária a instalação do pacote "shareutils" e do serviço "sendmail" em funcionamento.

Criou-se ainda um *script* de monitorização do estado dos processos do sistema para alertar e tentar relançá-los caso algum falhasse. De forma análoga aos outros *scripts*, foi colocado no ficheiro cron.hourly, sendo lançado de hora em hora. O envio de *emails* requer o serviço "sendmail":

```
#!/bin/sh
SERVICE='honeyd-int';
if ps ax | grep -v grep | grep $SERVICE > /dev/null
then
 echo ``$SERVICE service running, everything is fine''
else
 echo ``$SERVICE is not running''
 echo ``$SERVICE is not running!'' | mail -s ``$SERVICE ISEP falhou'' 1000261@isep.ipp.pt
  /usr/bin/honeyd -i eth1 -p /usr/share/honeyd/nmap.prints -x /usr/share/honeyd/xprobe2.conf
   -a /usr/share/honeyd/nmap.assoc -f /usr/share/honeyd/honeyd-int.conf -l
    /var/log/honeyd/honeyd-int.log 172.31.100.71-172.31.100.74 --disable-webserver
fi
SERVICE2='snort-int';
if ps ax | grep -v grep | grep $SERVICE2 > /dev/null
then
 echo ``$SERVICE2 service running, everything is fine''
else
 echo ``$SERVICE2 is not running''
 echo ``$SERVICE2 is not running!'' | mail -s ``$SERVICE2 ISEP falhou'' 1000261@isep.ipp.pt
  /usr/sbin/snort -de -l /var/log/snort-int -i eth1 -c /etc/snort/snort-int.conf
   --alert-before-pass --treat-drop-as-alert -F /etc/snort/excludes.conf -D
fi
SERVICE3='honeyd-ext';
if ps ax | grep -v grep | grep $SERVICE3 > /dev/null
then
 echo ``$SERVICE3 service running, everything is fine''
else
 echo ``$SERVICE3 is not running''
 echo ``$SERVICE3 is not running!'' | mail -s ``$SERVICE3 ISEP falhou'' 1000261@isep.ipp.pt
  /usr/bin/honeyd -i eth0 -p /usr/share/honeyd/nmap.prints -x /usr/share/honeyd/xprobe2.conf
   -a /usr/share/honeyd/nmap.assoc -f /usr/share/honeyd/honeyd-ext.conf -l
    /var/log/honeyd/honeyd-ext.log 193.136.62.71-193.136.62.74 --disable-webserver
fi
SERVICE4='snort-ext';
if ps ax | grep -v grep | grep $SERVICE4 > /dev/null
then
 echo ``$SERVICE4 service running, everything is fine''
else
 echo ``$SERVICE4 is not running''
 echo ``$SERVICE4 is not running!'' | mail -s ``$SERVICE4 ISEP falhou'' 1000261@isep.ipp.pt
/usr/sbin/snort -de -l /var/log/snort-ext -i eth0 -c /etc/snort/snort-ext.conf
 --alert-before-pass --treat-drop-as-alert -F /etc/snort/excludes.conf -D
fi
```

Foi configurada uma rotação de logs por tamanho dos ficheiros, sendo estes renovados quando atingiam 5MB. É necessário baixar a protecção do “SELinux” para que o sistema permita o novo lançamento dos serviços parados:

```
# /etc/logrotate.d/honeyd
# $Id$
/var/log/honeyd/honeyd-ext.log /var/log/honeyd/web.log
    rotate 5
    size 5000k
    postrotate
killall honeyd
  /usr/share/honeyd/start
    endscript
```

Para o automatismo do arranque do sistema foi criado um *script* de arranque para o Honeyd e outro para o Snort e colocados no ficheiro "rc.local". Estes limitam-se a chamar os programas com os parâmetros correctos. Apresenta-se os *scripts* para a implementação em CentOS por necessitarem também de lançar os serviços arpd:

```
#----- ARPD ---------
/usr/sbin/arpd 192.168.1.82
/usr/sbin/arpd 192.168.1.83
/usr/sbin/arpd 192.168.1.84
/usr/sbin/arpd 192.168.1.85
#----- HONEYD ---------
/usr/local/honeyd/bin/honeyd -p /usr/local/honeyd/share/honeyd/nmap.prints -x
 /usr/local/honeyd/share/honeyd/xprobe2.conf -a /usr/local/honeyd/share/honeyd/nmap.assoc
  -f /usr/local/honeyd/share/honeyd/honeyd.conf -l /var/log/honeyd/honeydlog
   192.168.1.82-192.168.1.85 --disable-webserver
#----------- SNORT -----------
/usr/sbin/snort -de -c /etc/snort/snort.conf --alert-before-pass --treat-drop-as-alert -F
 /etc/snort/excludes.conf -D
```

Foram necessárias medidas a nível de segurança física das máquinas, dado o facto de poderem colocar em risco a segurança das faculdades colaborantes. Desta forma houve o cuidado de apenas permitir conexões reais (não simuladas) ao sistema na porta 22 (SSH) e de um único endereço caso se tratasse de uma interface pública. Segue-se a configuração do serviço "iptables" implementado no Honeypot do ISEP:

```
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp -i eth0 -s 193.136.25.91 --dport 22 -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp -i eth1 --dport 22 -j ACCEPT
-A INPUT -j REJECT --reject-with icmp-host-prohibited
-A FORWARD -j REJECT --reject-with icmp-host-prohibited
COMMIT
```

4.5 Desenvolvimento da aplicação de preparação dos dados

A aplicação foi desenvolvida em Ruby 1.9.1. Esta linguagem foi escolhida por não necessitar de sistemas muito performantes para ser usada, oferecer potentes ferramentas de tratamento de *strings* e ficheiros e ser de código aberto. Tem como função principal ler dois ficheiros de *log*, um proveniente do programa Honeyd e outro do Snort, fundi-los e processar a informação resultante para poder ser submetida a algoritmos de *data mining*.

O algoritmo da aplicação é descrito de forma muito resumida na figura 4.2. Nesta são feitas alusões a BD Snort MySQL e BD Países Mysql. A primeira é uma base de dados criada em MySQL usada para guardar a informação oriunda do Snort, depois de filtrada e normalizada. É usada uma base de dados por forma a facilitar a pesquisa de registos quando se procura para uma dada entrada do Honeyd a respectiva entrada Snort. A BD Países MySQL foi retirada do site http://www.hostip.info e contém informação actualizada sobre o paralelismo entre redes de classe C públicas e os países onde foram atribuídas.

Outra nota referente à figura anteriormente mencionada diz respeito à operação “separar campos de acordo com o formato da linha”. Esta função verifica se o tipo de dados dos campos e o seu número corresponde a um conjunto de tipos de linha conhecidos inseridos previamente. O programa foi sendo aperfeiçoado por forma a conhecer todos os formatos de linha válidos.

Além deste controlo, é também feito o filtro de registos consoante o IP de origem e destino. O tráfego proveniente ou dirigido a máquinas de gestão do *honeypot* ou monitorização da rede é excluído, assim como o que é dirigido a IPs que não pertencem a nenhum dos *honeypots* dado que não serão úteis para a análise. É demonstrado na tabela 4.1 quais os campos de entrada na aplicação e quais os de saída. Nos anexos C,D e E encontra-se um excerto de um ficheiro de *log* do Honeyd e do Snort, assim como de um ficheiro resultante da aplicação de preparação de dados.

Esta aplicação corre em modo de texto e recebe como parâmetros o caminho para o ficheiro de *log* do Honeyd, o caminho para o ficheiro de *log* do Snort, o caminho para o ficheiro CSV de saída, os IPs das máquinas de gestão do *honeypot* e os IPs dos sistemas de monitorização da rede.

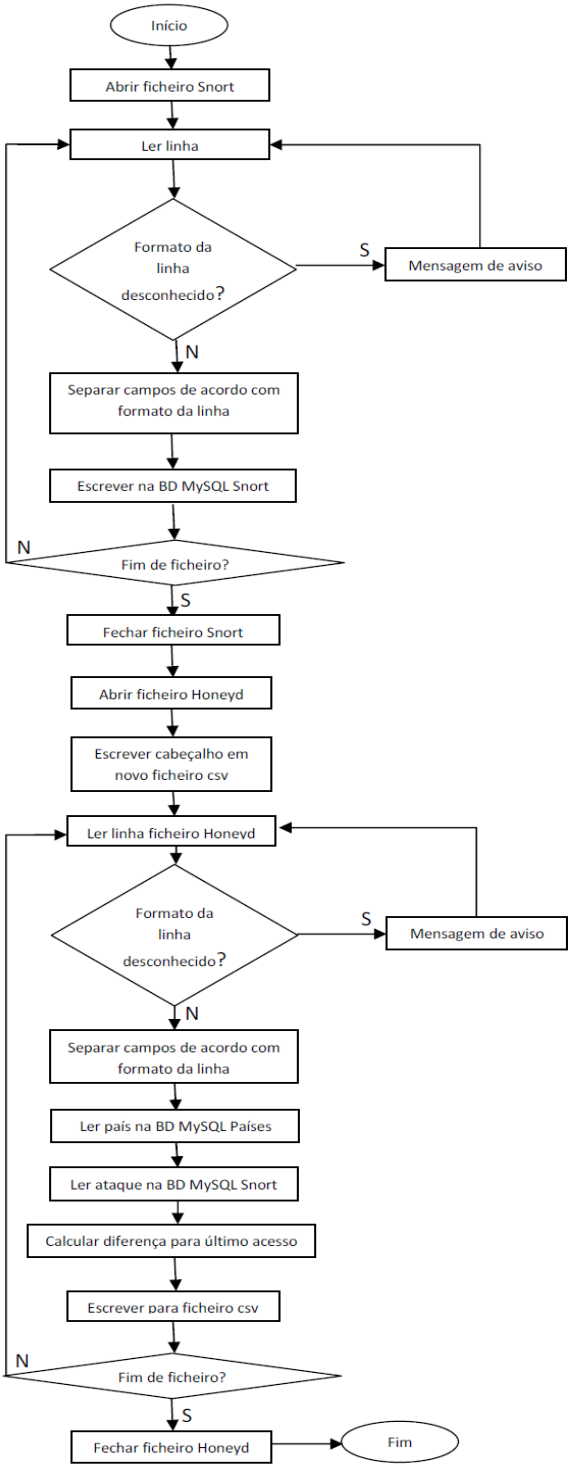

Figura 4.2: Algoritmo da aplicação desenvolvida para preparação dos dados recolhidos

Tabela 4.1: Campos que resultam da aplicação de tratamento de dados

Campos Snort	**Campos Honeyd**	**Campos usados para a fusão**	**Campos do ficheiro resultado**
ataque	inicio_fim		ataque
classificação	servidor		servidor
prioridade	so_servidor		so_servidor
data	data	sim	data
hora	hora	sim	hora
type	so_origem		so_origem
len	pais		pais
ip_origem	ip_origem	sim	ip_origem
porto_origem	porto_origem		porto_origem
ip_destino	ip_destino	sim	ip_destino
porto_destino	porto_destino		porto_destino
protocolo	protocolo	sim	protocolo
ttl			ultimo_acesso
tos			
id			
iplen			
dgmlen			

4.6 Resultados obtidos

A solução implementada mostrou-se eficaz. Embora alguns *honeypots* tenham bloqueado no início da implementação devido a pedidos em massa, provavelmente ataques de DoS, esta situação foi controlada com implementação de *timeouts* nos *scripts* de emulação dos serviços. Desta forma, a resposta a um pedido de um atacante tinha um período de tempo definido no qual ele podia interagir até a ligação cair, em vez de aguardar indefinidamente como se estava a verificar. Os diversos sistemas de protecção contra falhas e tratamento dos *logs* funcionaram muito bem. Os sistemas funcionaram durante aproximadamente meio ano sem paralisações, excepto duas situações pontuais devido a corte de corrente eléctrica.

O equipamento disponibilizado foi suficiente para a solução. Esta demonstrou-se pouco exigente em recursos. Apenas se salienta como requisito alguma capacidade de processamento que é necessária quando as máquinas recebem muitos pedidos em simultâneo. A utilização de plataformas de virtualização foi considerada ideal pois permite o acesso remoto aos computadores e possibilidade de gestão total, desde reiniciar uma máquina até à alteração da sua constituição a nível de recursos. As versões de software mais recentes mostraram-se mais funcionais para este tipo de implementação, como foi o caso da distribuição Linux Fedora12.

Entre a configuração de interfaces de rede virtuais ou recurso à ferramenta arpd não houve uma preferência. Da mesma forma que os interfaces virtuais obrigam a configuração adicional e definição

de rotas, o arpd deu problemas de instalação em alguns casos por incompatibilidade com outras bibliotecas.

A utilização do Snort juntamente com o Honeyd considerou-se importante e bastante funcional pois foi capaz de identificar bastantes ataques. O volume de dados recolhido e a sua análise serão referidos mais à frente. Na maioria dos casos, após a instalação das máquinas, a ocorrência de ataques não demorava mais de três horas. Isto demonstra que as máquinas ligadas à Internet são sujeitas diariamente a diversas ameaças e que as faculdades, em particular, são um alvo muito procurado.

A ferramenta desenvolvida para preparação dos dados foi muito bem testada antes de ser usada em produção. Este tempo foi recompensado pelo bom funcionamento verificado quando foi necessário tratar os dados para serem submetidos ao programa de *data mining*. Esta fase é descrita na secção 6.1 do capítulo 6.

Capítulo 5

Metodologias de mineração de dados em *logs*

5.1 *Data mining*

Tal como foi referido anteriormente, pretende-se analisar os dados recolhidos pelos *honeypots* instalados e tentar obter conhecimento aplicando-lhes técnicas de *data mining*. *Data mining* ou mineração de dados é uma tecnologia que se tem vindo a tornar cada vez mais importante no campo de grandes bases de dados e armazéns de dados. A analogia é feita com base nos grandes pedaços de terra brutos que são extraídos das minas e que, após serem submetidos a vários processos, levam à descoberta de pequenas porções de material precioso. Da mesma forma, em mineração de dados uma grande quantidade de dados é processada para construir um modelo simples e útil. Mas este campo não é só um problema de bases de dados. Faz também parte do campo da inteligência artificial. Para ser inteligente, um sistema que se encontra num ambiente dinâmico deve ter capacidade de aprender e se adaptar [Alpaydin, 2004].

Também é referida como descoberta de conhecimento, pois a sua implementação permite descobrir conhecimento pouco óbvio, implícito, anteriormente desconhecido e potencialmente útil. Conhecimento que, pela simples análise visual ou estatística seria potencialmente impossível de descobrir [Hong et al., 2009].

A aplicação de técnicas de *data mining* utiliza-se habitualmente em grandes corporações que registam toda a sua informação em armazéns de dados e, posteriormente, a analisam com vista a apoiar o alcance do seu objectivo. Entende-se como armazém de dados (*data warehouse*) uma grande base de dados ou conjunto de bases de dados que contêm toda a informação de todas as áreas de uma organização. Tenta encontrar relacionamentos entre os dados que possam servir para prever situações com um determinado grau de incerteza. Estes relacionamentos são descritos em modelos cuja finalidade é muito abrangente. Devido à utilização de armazéns de dados, os resultados obtidos com estas técnicas conseguem conjugar todas as componentes de um negócio ou actividade. Tomando como exemplo um

Hospital, estas metodologias podem ser usadas desde o apoio à decisão da medicação com base na análise dos dados médicos, até à distribuição dos doentes assente na criação de perfis [Veerasamy et al., 2009].

Genericamente existem duas abordagens de mineração de dados: preditiva e descritiva. A visão preditiva permite prever resultados, ou seja, tentar adivinhar o comportamento ou decisão de um determinado elemento. Por outro lado, a análise descritiva permite perceber informação que não era perceptível à vista desarmada e ajudar a compreender a situação em que um determinado processo se encontra. Resumidamente, a análise preditiva permite ter noções sobre o futuro, enquanto que a descritiva se refere ao presente e passado [Chuvakin, 2005].

A cada uma destas abordagens estão associadas várias técnicas de descoberta de conhecimento. No caso da análise descritiva existe: *clustering* que consiste em agrupar os registos consoante a semelhança das suas características; regras de associação que permitem determinar relações de associação entre os atributos de uma entidade, sessão ou grupo, inferindo regras do género 'quem vai a uma loja e compra pão, compra também leite'; elementos frequentes que identificam registos que se repetem frequentemente, apresentando apenas pequenas variações, como por exemplo a nível temporal; e padrões sequenciais que consistem em identificar sequências de registos que se repetem frequentemente, apresentando apenas pequenas variações, como por exemplo a nível temporal.

Para a abordagem preditiva conhece-se: classificação que consiste em agrupar os registos por classes, ou seja, de acordo com a aplicação de condições aos seus atributos; regressão que permite prever o comportamento de elementos semelhantes aos que estão a ser minerados; e o desvio padrão que é capaz de identificar se os dados correspondem ou não a um determinado perfil [Chuvakin, 2005].

5.2 *Logs*

Considera-se *log* o registo de eventos ocorridos num sistema, independentemente de se tratar de um programa, um serviço, ou um dispositivo. A informação registada pode ser referente ao estado actual do sistema, erros de funcionamento, sucesso de operações, pedidos de acesso, autorizações e bloqueios.

Hoje em dia os servidores de uma instituição recebem mais tráfego do que alguma vez sucedeu e isto leva, naturalmente, a um aumento do número de eventos registados. Desta forma, é cada vez mais difícil a um administrador de sistemas e/ou rede identificar que informação de *log* é efectivamente útil.

Esta problemática surge porque quando um servidor é configurado para registar os acessos que lhe são efectuados guarda um conjunto muito abrangente de informação que tanto pode ser referente a uma utilização normal dos serviços disponibilizados como a tentativas de ataque. Esta dificuldade vai aumentando à medida que a quantidade de informação registada aumenta. Por outro lado, pode até não existir nenhum registo referente a ataques ou relevante para a sua detecção [Chuvakin, 2005].

Outra questão importante é o grau de conhecimento sobre os eventos registados que é exigido a quem faz a análise de registos de eventos. A detecção de padrões ou registos relevantes implica um conhecimento profundo do funcionamento dos sistemas, da rede em que estão implementados e até

dos seus utilizadores [Chuvakin, 2005].

Face a estas dificuldades, surge a necessidade de soluções de tratamento de *logs* capazes de identificar informação relevante e apoiar a melhoria de funcionalidade e segurança dos sistemas.

5.3 Mineração de dados em *logs*

Com o intuito de identificar informação relevante em registos de eventos, tem-se usado mineração de dados com diferentes abordagens consoante o tipo de conhecimento que se pretende obter. Isto deve-se à importância crescente do registo de eventos na administração de sistemas e redes, e a sua necessidade para a melhoria do desempenho das tarefas do administrador [Vaarandi, 2004].

Das diferentes abordagens de mineração de dados, a que se pretende é um pouco invulgar. Procura-se encontrar conhecimento que permita conhecer os atacantes das instituições de ensino superior e a forma como actuam. Dado que não há muita informação sobre esta perspectiva, fez-se uma analogia a três vertentes: *web mining*, detecção de ataques e KDD.

5.3.1 *Web mining*

Web mining tem por objectivo procurar associações em informação da Internet (Web) e pode-se dividir em três tipos, *web content* (conteúdo *web*), *web structure* (estrutura *web*) e *web usage* (utilização web).

Web content mining permite descobrir informação útil a partir do conteúdo da *web* [Morais, 2007]. É um processo automático que vai além de extracção de palavras-chave. Dado que o conteúdo de um documento de texto não apresenta semântica passível de ser interpretada por máquinas, algumas abordagens sugerem a reestruturação do conteúdo do documento para uma representação que pudesse ser explorada por máquinas. A técnica habitual é o uso de *wrappers*[1] para mapear documentos para um modelo de dados. Existem duas estratégias para mineração de conteúdo web, as que descobrem directamente o conteúdo dos documentos e as que melhoram a pesquisa de conteúdo em documentos por parte de outras ferramentas, como motores de busca [Freisleben and Galeas, 2009].

Web Structure Mining permite descobrir informação útil a partir da estrutura de ligações [Morais, 2007]. A Internet pode revelar mais informação do que aquela que está presente nos documentos. Por exemplo, os *links* que apontam para um documento indicam a popularidade do documento enquanto que os *links* que acompanham um documento indicam a riqueza ou variedade de tópicos abordados no documento. Isto pode ser comparado com citações bibliográficas. Quando um artigo é citado variadas vezes torna-se importante. O PageRank[2] e métodos CLEVER[3] tiram partido deste tipo de informação

[1] Neste caso refere-se a *scripts* que criam modelos de dados que definem documentos web, com base no seu varrimento prévio linha a linha.

[2] Sistema que atribui diferentes níveis de importância a páginas de Internet. Desenvolvido pelos fundadores do Google e usado no seu motor de busca [Google, 2010].

[3] IBM CLEVER project - algoritmo baseado em links, alternativo ao PageRank [Paarl, 2010].

contido nos *links* para encontrar páginas pertinentes. Por meio de contadores de *hyperlinks* conseguem uma percepção resumida dos conteúdos de um site [Freisleben and Galeas, 2009].

Web Usage Mining permite descobrir informação útil a partir de dados referentes a acessos a páginas *web* [Morais, 2007]. Os servidores *web* guardam e acumulam informação sobre as interacções com o utilizador sempre que é efectuado um pedido para um determinado recurso. Analisando registos correspondentes a sítios *web* diferentes pode ajudar a perceber o comportamento do utilizador e a estrutura web. Esta técnica tem duas tendências principais, *general access pattern tracking* (identificação de padrões gerais de acesso) e *customized usage tracking* (identificação personalizada de utilização). A primeira, analisa os *logs* para perceber padrões de acesso e tendências. Isto pode trazer benefícios a nível de estruturação e disposição de conteúdos. Na segunda, a análise tem por objectivo identificar tendências individuais, personalizando os sítios *web* para os seus utilizadores, de acordo com os seus padrões de acesso [Freisleben and Galeas, 2009].

A analogia mais próxima seria mineração de utilização *web* dado que, há semelhança do que se pretende, parte de um conjunto de registos de eventos de acessos e tem por objectivo caracterizar os acessos em si. Nesta vertente tem sido mais usada a mineração de regras de associação com recurso ao algoritmo Apriori ou variantes deste. Habitualmente, a mineração é feita sobre *clusters* derivados da aplicação do algoritmo K-means ou variantes [Khribi et al., 2009].

5.3.2 KDD

KDD - *Knoledge Discovery from Databases* (Descoberta de Conhecimento em Bases de Dados) consiste em identificar informação coerente e relevante em grandes bases de dados [Frawley et al., 1991]. Apesar de existir muitas metodologias aplicadas no seu estudo, nomeadamente regras de classificação, *cluster* e padrões sequenciais, a mais utilizada tem sido a extracção de regras de associação [Agrawal and Srikant, 1994; Agrawal et al., 1997; Han and Fu, 1995; Mannila et al., 1994; Park et al., 1997; Savasere et al., 1995; Srikant and Agrawal, 1995]. São usadas para descrever a correlação entre itens e conjuntos de itens e têm aplicação prática em várias áreas [Hong et al., 2009].

A mineração de regras de associação é muito frequente entre as técnicas usadas em Descoberta de Conhecimento em Bases de Dados. Tem por objectivo descobrir relações entre itens e conjuntos de itens. Em geral, pode ser divido de uma forma muito simplista em duas tarefas [Hong et al., 2009]: encontrar conjuntos de itens frequentes satisfazendo um suporte mínimo definido pelo utilizador, a partir dos dados; e gerar regras de associação interessantes satisfazendo um valor mínimo de confiança estipulado pelo utilizador, a partir dos conjuntos de itens frequentes encontrados anteriormente.

Partindo do exemplo de uma caixa registadora de uma loja, a partir da mineração de regras de associação é possível encontrar-se associações entre os produtos que cada cliente compra. Se os clientes que compram o produto X também adquirem na mesma compra o produto Y, caso haja um cliente que comprou X e não comprou Y este é um potencial consumidor do produto Y. Uma vez encontrado este tipo de cliente pode-se tentar direccioná-los neste sentido [Alpaydin, 2004]. No presente contexto, a geração de regras de associação pode ajudar a perceber associações do género quem ataca um servidor *web* Windows ataca também o servidor de *email* Windows. Caso o administrador de sistemas note

que um servidor *web* Windows esteja a ser atacado pode tomar precauções relativamente aos outros servidores Windows, por exemplo, cortando todas as ligações com o endereço IP do atacante. Estas regras podem conter vários atributos sobre a mesma entidade, por exemplo, quem ataca um servidor *web* Windows e acede a partir da China, irá atacar todos os servidores *web* da mesma rede.

Neste sentido, têm sido propostos muitos algoritmos baseados no Apriori [Vaarandi, 2004; Hong et al., 2009]. Segundo este, quando a percentagem de transacções que contêm um conjunto de itens candidato é maior ou igual a um mínimo especificado pelo utilizador, denominado suporte, esse conjunto é considerado como frequente e seus itens como correlacionados entre si [Hong et al., 2009].

Outra metodologia muito usada é a mineração de padrões temporais que faz uma análise temporal dos dados e procura encontrar regularidades e padrões. Enquanto que a mineração de regras de associação permite descobrir correlações não ordenadas entre os itens de uma dada base de dados, o *data mining* temporal permite descobrir correlações ordenadas [Hong et al., 2009].

Análise temporal pode ser efectuada com base em associações sequenciais ou periódicas [Li and Deogun, 2005], tais como as associações cíclicas (onde as regras associativas ocorrem durante um intervalo regular de tempo) [Ozden et al., 1998] e de calendário (onde as regras associativas ocorrem durante determinadas datas, podendo não ser regulares, por exemplo uma regra que se verifica quando é feriado) [Li et al., 2003]. Tem atraído muitos investigadores da actualidade. Consiste na análise de dados temporais e descoberta de padrões temporais e correlação ordenada de itens ao longo do tempo. Considerando como exemplo uma venda de gelados, ao analisar a sua informação de vendas certamente se ia obter uma correlação entre um aumento de vendas e as épocas mais quentes do ano (Verão). Este tipo de descoberta só pode ser conseguido caso se defina convenientemente a janela temporal para o processo de mineração [Roddick and Spiliopoulou, 2002]. Por outro lado, uma definição estática da janela temporal pode também impedir a descoberta de outras informações sobre os itens [Hong et al., 2009].

5.3.3 Detecção de intrusões

A detecção de intrusões (*intrusion detection*) tem por objectivo identificar e caracterizar ataques. É a área mais desenvolvida em termos de aplicação de mineração de dados a *logs* e divide-se em dois grandes tipos: detecção de abuso (*misuse*) e anomalias (*deviation*). Abuso refere-se a detecção de ataques baseada em padrões extraídos de ataques conhecidos. Detecção de anomalias consiste em identificar intrusões baseando-se em desvios significativos da actividade considerada normal. Detecção de abuso apresenta normalmente uma baixa taxa de falsos positivos mas não consegue detectar novos ataques. Por outro lado, a detecção de anomalias pode detectar ataques desconhecidos mas apresenta vulgarmente uma elevada taxa de falsos positivos. Têm sido propostas várias abordagens para combinação destas duas técnicas de detecção [Zhang et al., 2008b].

Relativamente à presente abordagem encontra-se paralelismo com a detecção de abuso, dado que não requer processamento em tempo real, permitindo que os dados sejam guardados e trabalhados a posteriori. Para este tipo de abordagem com recurso a técnicas *data mining* tem sido mais usada a

classificação em conjunto com *clustering* [Thonnard and Dacier, 2008; Li and Chandra, 2007; Zhang et al., 2008b].

Classificação permite ao analista humano identificar não só o tráfego que corresponde a um ataque mas também a razão desta classificação [Grégio et al., 2007]. Tenta-se atribuir cada item a uma categoria ou classe, tendo por base um modelo criado a partir de dados pré-classificados usados para treino, tratando-se portanto de aprendizagem supervisionada [Schenker et al., 2005]. É usada, por exemplo, pelas instituições financeiras. Consoante o nome, salário, quantia bancária, profissão, idade, histórico financeiro, entre outros, a instituição é capaz de calcular o risco associado para um potencial cliente e, consequentemente, prever se é ou não um cliente de risco. Para isto é usado um modelo que vai sendo treinado com informação passada de outros clientes, nomeadamente se foram ou não bons pagadores. Após introdução dos dados de um novo cliente ele classifica-o numa de várias classes. Neste caso poderia haver apenas duas, alto-risco e baixo-risco [Alpaydin, 2004]. Relativamente ao presente contexto, um modelo de classificação pode prever que um acesso proveniente da China a determinadas horas seja um ataque do tipo X. Caso a prevenção contra o ataque seja muito dispendiosa em termos de processamento, o administrador de sistemas pode tomar precauções, informando-se sobre o tipo de ataque, e implementá-las apenas nas horas de maior risco. Os algoritmos mais utilizados nesta área são algoritmos genéticos e redes neuronais [Zhang et al., 2008b; Veerasamy et al., 2009; Grégio et al., 2007]. No âmbito de *clustering* nota-se clara preferência pelo algoritmo K-means [Thonnard and Dacier, 2008].

Clustering é usado com o objectivo de separar um dado conjunto de dados em grupos, de tal forma que os itens do mesmo grupo sejam semelhantes entre si e dissimilares dos itens dos outros *clusters*. A aprendizagem considera-se não supervisionada dado que não são fornecidos aos algoritmos dados pré-classificados para treino [Schenker et al., 2005]. No presente contexto tem sido usado com intuito de perceber associações entre a localização geográfica dos acessos e o tipo de ataque, compreendendo muitas vezes o motivo do ataque. Pode-se ainda detectar o grau de desconfiança de uma *subnet*. "Clientes comprometidos têm tendência para pertencer a redes promiscuas, principalmente máquinas *zombie* pertencentes a *botnets*" [Collins et al., 2007]. Este tipo de informação pode ajudar a prevenir determinados ataques e reduzir a exposição de uma rede.

Costumam ser usadas em conjunto, *clustering* numa fase inicial em que se criam conjuntos constituídos por elementos com atributos aproximados, como por exemplo identificador de sessão próximo ou endereços IP de origem e destino iguais. Depois, a cada um dos *clusters* , aplica-se algoritmos de classificação[Veerasamy et al., 2009]. Este estudo pode levar a conclusões relativamente à localização geográfica dos acessos, se há algum tipo de acesso específico apenas oriundo de um conjunto restrito de países, identificação de subnets mais propícias a serem origem de acessos ilícitos, períodos em que há mais acessos ou identificar o mesmo tipo de acesso sempre no mesmo horário tipicamente devido a *worms* [Thonnard and Dacier, 2008].

A detecção destes acessos frequentes espaçados no tempo é o principal objectivo da análise temporal, outro método muito usado nesta área, pois pode providenciar informação útil para compreender o ataque, assim como pode ser usado como um parâmetro do *clustering*. Neste caso, a escolha da

granularidade do tempo vai influenciar o tipo de ataque que se pode identificar. Caso os intervalos de tempo usados sejam pequenos, de uma hora por exemplo, pode-se descobrir ataques do tipo *botnets* ou *flash worms*. Outro tipo de *worms* que use um esquema de propagação mais lento só será detectado com uso de uma escala temporal maior [Thonnard and Dacier, 2008]. Por exemplo, a detecção de actividade de um padrão recorrente diurna/nocturna pode indicar um esquema de propagação de um *botnet* [Dagon et al., 2006]. Outras análises podem também ser relevantes para aprendizagem dos processos de ataque e do modo de operação dos atacantes, tais como a resolução de nomes das máquinas atacantes ou conhecimento prévio das características de determinado *malware* [Thonnard and Dacier, 2008].

5.4 Definição das metodologias a implementar

Pretende-se conjugar as técnicas de mineração de dados que têm vindo a ser usadas nestas três abordagens diferentes de tratamento de *logs*: *web* mining, detecção de intrusões e descoberta de conhecimento em bases de dados. Espera-se conseguir obter informação que possa ajudar um administrador de sistemas e/ou redes a defender-se, quer no campo da previsão de ataques, quer na sua reacção a estes, à semelhança dos resultados obtidos por um estudo que demonstrou que alguns *worms* mostram claramente nos logs o seu método de propagação, como por exemplo uma tendência para procurar máquinas próximas ou da mesma rede, de forma a optimizar a sua infecção [Chen et al., 2003]. Deste modo, analisando os logs de ataque quanto à distância entre os IPs dos atacantes e os das suas vítimas, era possível reconhecer o modo de propagação de certas famílias de *malware* [Thonnard and Dacier, 2008] e assim prevenir os seus ataques.

Um ataque comum detectado na análise de *logs* é o DOS (*Denial of Service*) através de “ping”, tráfego ICMP dirigido aos servidores em quantidade exagerada, com o intuito de os ocupar por forma a não terem capacidade para responder aos clientes [Chuvakin, 2005]. Pode também ser apurado o grau de perigo de ataques internos, ou seja, por parte de utilizadores da rede interna.

Outra análise interessante é a procura de anomalias frequentes. Por exemplo, múltiplos ataques provenientes do mesmo endereço IP, da mesma subnet, ou o mesmo tipo de ataque ocorrido repetidamente [Beheshti et al., 2008]. Num estudo feito no Consolado de Pesquisa Científica e Industrial de África do Sul (CSIR), descobriu-se que a sua rede era atacada principalmente durante a tarde e por residentes daquele país, levando a suspeitar de estudantes que, depois de saírem da escola, iam para casa e procuravam vulnerabilidades nos computadores vizinhos, voluntariamente ou não. Foram também detectados muitos ataques provenientes de uma empresa comunicações Asiática e de outra Coreana. Estes ataques tinham como objectivo explorar vulnerabilidades SMTP. A Ásia é conhecida pelas notórias práticas de SPAM e, provavelmente, este foi mais um destes casos [Veerasamy et al., 2009]. Este é mais um exemplo do que se espera obter a partir dos registos de eventos referentes a tentativas de acesso ilícitas a Instituições de Ensino Superior, efectuados no âmbito desta dissertação.

No próximo capítulo descreve-se em detalhe a implementação dos algoritmos Apriori, K-means, Redes Neuronais e Análise Temporal. Estes foram considerados os mais usados nesta área e os mais

indicados para o presente contexto. Para esta análise utilizou-se a aplicação SPSS Clementine 12.0 e os dados recolhidos nas cinco instituições de ensino superior portuguesas que aceitaram colaborar. Os resultados desta experiência são também descritos e interpretados no capítulo seguinte.

Capítulo 6

Mineração dos dados obtidos

6.1 Descrição do processo adoptado

No capítulo anterior refere-se os algoritmos Apriori e conjugação K-Means com Apriori como os mais utilizados em mineração de *logs*. Realça-se também Redes Neuronais e Identificação de Padrões Sequenciais.

Para aplicação destas metodologias, com vista à descoberta de conhecimento sobre atacantes de instituições de ensino superior e a sua forma de actuar, seguiu-se o processo de descoberta de conhecimento aplicado a *logs* que consiste nos seguintes passos [Singhal and Jajodia, 2006]:

1. Limpeza de dados: remover dados irrelevantes ou com erros
2. Integração de dados: combinar dados de diferentes fontes
3. Transformação dos dados: transformação da informação e derivação de campos
4. Mineração de dados: aplicar metodologias de *data mining* para encontrar relacionamentos nos dados
5. Avaliação e apresentação: apresentação e interpretação dos resultados

Após recolha de *logs* das várias implementações de Honeyd e Snort, obteve-se um ficheiro de texto para cada uma destas aplicações, por cada faculdade colaboradora. Estes ficheiros foram submetidos a uma aplicação desenvolvida no âmbito desta dissertação, em linguagem Ruby, que além de fundir a informação registada pelo Honeyd com a que foi capturada pelo Snort, elimina tráfego impróprio para análise (como acessos de gestão, monitorização ou actualização), deriva o país de onde é feito o ataque a partir do endereço IP utilizado para o acesso, calcula o espaçamento temporal entre os dois últimos acessos de um determinado atacante e uniformiza o ficheiro resultante, tornando-o legível por qualquer programa que suporte o formato CSV.

Cada linha do ficheiro resultante contém informação referente a um acesso, nomeadamente: IP de origem e destino, porta de origem e destino, sistema operativo de origem e destino, serviço atacado,

país do atacante, data, hora, protocolo, nome e intervalo do ataque. Os restantes campos foram considerados desprezáveis, dado que não podiam ser usados em mecanismos de prevenção nem seriam úteis à reacção a ataques informáticos. Na secção 4.5 do capítulo 4 encontra-se informação detalhada sobre o desenvolvimento desta aplicação.

Com intuito de melhorar a utilidade dos dados em termos da sua mineração e com auxílio da aplicação SPSS Clementine 12.0, derivou-se o campo subnet, que corresponde aos primeiros dois bytes do IP de origem. Discretizou-se a hora por classes a que se chamou “altura” correspondentes a altura do dia a que a hora diz respeito: manha (08:00 - 13:00), tarde (13:00 - 20:00), noite (20:00 - 01:00) e madrugada (01:00 - 08:00). A partir da data, derivou-se o campo “dia da semana”, um valor numérico inteiro compreendido entre 1 e 7, sendo o 1 “Domingo”, o 2 “Segunda-Feira”, terminando no 7 “Sábado”. Estes campos foram derivados com vista à identificação de padrões temporais cíclicos.

Derivou-se também um campo que contabiliza o número de acessos efectuado em cada sessão, sendo esta definida pelo conjunto chave: data, IP de origem e sistema operativo de origem. Foi também calculado o número de vezes que um determinado espaçamento temporal entre os dois últimos acessos de um atacante foi repetido, com o intuito de identificar elementos sequenciais frequentes.

O campo sistema operativo do atacante foi discretizado porque se verificou que a detecção de sistema operativo funcionava bem para a família do sistema (windows, linux, mac), mas não para a versão (XP, Vista, 7). Desta forma, os valores foram convertidos por forma a agregar todas as versões do mesmo sistema operativo, passando a ser um dos seguintes: windows, linux ou outros.

Por fim, discretizaram-se os campos portas de origem e destino. As portas de origem foram divididas em classes que correspondem a intervalos de valores, por forma a que todas as classes tenham um número equilibrado de elementos. O campo porta destino separou-se em dois grupos apenas, onde um corresponde às portas que estão abertas e que os serviços implementados usam habitualmente, e outro que engloba acessos a qualquer outra porta que, por norma, só seria usada por aplicações específicas (nomeadamente programas de ataque a sistemas informáticos).

6.2 Resultados obtidos

A análise dos dados foi efectuada com auxílio da aplicação SPSS Clementine 12.0. Como todos os dados correspondem a ataques, surgiu como dificuldade identificar quais as regras que teriam utilidade num contexto real, com a presença de tráfego lícito e ilícito. Além desta distinção, é necessário também que as regras evidenciadas tenham relevância prática nas tarefas de um administrador de sistemas e/ou redes, ou auxiliando a reacção a ataques ou a sua detecção. Como exemplo refere-se a regra “Se porto destino é comum então será iniciado um ataque do tipo ’ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited’ ”. Esta informação permite perceber que os ataques são feitos usando portas comuns e que precisam de estar “abertas” para o normal funcionamento dos serviços. Contudo, não tem qualquer relevância em termos de prevenção ou reacção, dado que todo o tráfego seguro será feito por estas mesmas portas de destino.

Perante várias regras semelhantes, onde um elemento coexistente com outros dá origem à mesma

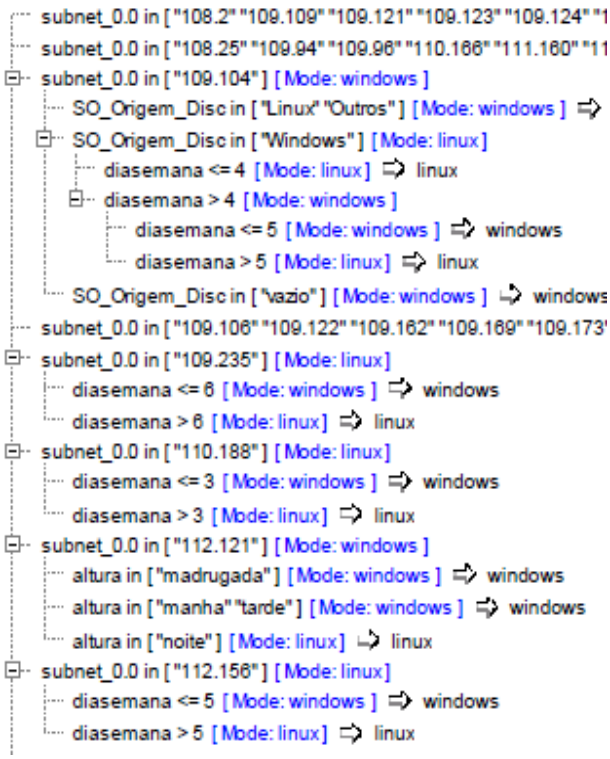

Figura 6.1: Exemplo de resultados obtidos com o algoritmo C5.0

ocorrência, optou-se pela condição mais abrangente. Tomando como exemplo as regras, se o ataque é "ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited" e o protocolo TCP, o servidor alvo será *web*, e se o mesmo for alvo independentemente do protocolo usado, a regra realçada seria: ataque "ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited" dá origem a alvo "servidor *web*", ignorando o protocolo.

No que se refere ao apuramento de resultados através da ferramenta SPSS Clementine 12.0, foi conseguida a extracção de conhecimento perante a aplicação do algoritmo de classificação C5.0, ao contrário da execução do algoritmo Redes Neuronais. Contudo, a sua demonstração de resultados não foi incluída dada a sua especificidade e tamanho. Estes algoritmos retornam modelos na forma de árvores de decisão que, tal como referido no capítulo anterior, podem prever ataques com uma determinada taxa de erro estimada. Dado que os modelos obtidos foram bastante extensos e que a sua aplicação é condicionada pelo seu objectivo e ambiente, optou-se por descrever apenas o tipo de modelos obtido. Estes consistiram em esquemas derivados da seguinte forma: perante um determinado conjunto de sub-redes, países de origem dos acessos, nomes de ataques e portas de origem e destino prevê-se o conjunto serviço e sistema operativo que irá ser atacado. Na figura 6.1 pode-se ver um exemplo de resultados obtidos com o algoritmo C5.0.

Os métodos com que se obteve resultados mais interessantes foram as regras de associação e a

sua conjugação com *clustering*. O número de *clusters* usado em cada caso prático foi definido com base em diversas experiências, tendo sido escolhida a situação que apresentava maior equilibro entre uma maior quantidade e diversidade de valores em cada *cluster*. Em seguida será feita uma descrição e interpretação dos resultados obtidos através da implementação do algoritmo Apriori e a sua aplicação em conjunto com o algoritmo K-Means. Acrescentou-se alguma informação estatística considerada interessante.

Contudo, a análise efectuada nas secções seguintes é apenas referente a dados recolhidos por *honeypots* externos. Os *honeypots* privados ou internos das várias instituições analisadas, que ao contrários dos públicos apenas estiveram expostos às redes locais de cada instituição, foram acedidos poucas vezes e por poucas máquinas. Por isso, a informação que recolheram não é útil para análise e demonstra que estas instituições não são ameaçadas pelos computadores ligados às suas redes internas.

6.2.1 ISEP - servidor público (de 19/12/2009 a 18/05/2010)

Através de análise estatística (mera observação):

- Destacou-se o dia 31-12-2009 com 25% dos acessos, onde 97% dos servidores atacados são de email, e a maioria dos ataques ocorreu entre as 21h05m e as 21h20m através do IP 65.106.230.146 registado na Flórida(16h05m e 16h20m hora local). Foi retirado este conjunto dia e IP do resto da análise para evitar a sua influência nos resultados. Restaram para análise 118664 registos, dos 157635 iniciais.
- O protocolo mais utilizado foi TCP. UDP e ICMP foram pouco usados.
- 57% dos ataques foi lançado a sistemas Windows e 59% a servidores *web*.
- Os portos mais atacados foram: 445, 8335, 135, 80, 22.
- Embora em apenas 41% dos acessos tenha sido possível registar o sistema operativo de origem, destacou-se o Windows XP com 32%.
- Nota-se que a maioria dos acessos ocorreu durante a tarde.
- A maioria dos acessos foi, ordenado do mais frequente para o menos, proveniente dos EUA, Portugal e China.
- Dos IPs de origem destacaram-se 2 que juntos participaram em 8% dos acessos:
  - 193.136.2.225 - videoconf03.fccn.pt
  - 66.186.59.50 - ircu.krypt.com (Califórnia - EUA) Esta máquina é conhecida na Internet pelos seus portscans mas desconhece-se o seu verdadeiro intuito.

- Além dos 2 IPs principais referidos anteriormente realça-se a máquina 193.136.33.211 - quimica-p-101.fe.up.pt pertencente à Faculdade de Engenharia da Universidade do Porto (FEUP). Foi registado um elevado número de acessos provenientes da FEUP.
- O tipo de ataque identificado mais comum foi “ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited”.
- A maioria dos ataques identificados como recorrentes foi espaçada no tempo em 1 (1286 ocorrências), 2 (574 ocorrências) e 10 horas (421 ocorrências).

Através do algoritmo Apriori:

- Se ataque = “ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited” -> porta destino = invulgar (78% de suporte e 90% de confiança). Se porta destino = invulgar -> ataque = “ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited” (87% de suporte e 81% de confiança).
- Se ataque = “ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited” -> número de acessos < 50 (80% de suporte e 67% de confiança). Se porta destino = invulgar -> número de acessos < 50, (87% de suporte e 70% de confiança).
- Se ataque = “ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited” -> servidor atacado = web (78% de suporte e 61% de confiança).
- Filtrando os dados considerando apenas a rede “66.186.0.0/16”: se ataque = “ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited” -> porto de origem > 5000 e < 10000, e vice-versa, (100% de suporte e confiança).
- Filtrando os dados considerando apenas a rede “66.186.0.0/16”: se número de acessos > 50 -> ataque = “ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited” (87% de suporte e 100% de confiança). Se ataque = “ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited” -> número de acessos > 50 (100% de suporte e 87% de confiança).
- Filtrando os dados considerando apenas a rede “193.136.0.0/16”: se protocolo = TCP e porto de destino = invulgar -> ataque = “ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited”, (67% de suporte e 100% de confiança)
- Filtrando os dados considerando apenas a rede “193.136.0.0/16”: se porto de origem < 5000 -> ataque = “ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited” (62% de suporte e 100% de confiança).
- Filtrando os dados considerando apenas a rede “193.136.0.0/16”: se porto origem < 5000 e protocolo = TCP, máquina atacante = windows (61% de suporte e 97% de confiança).

- Filtrando por acessos a servidores *web*: se ataque = “ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited” -> servidor atacado = windows (87% de suporte e 61% de confiança).

- Filtrando por acessos portugueses: se rede = 193.136.0.0 -> ataque = “ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited” (87% de suporte e 69% de confiança).

Através de Apriori com dados obtidos do maior cluster resultante da aplicação do algoritmo K-Means para 3 *clusters*, com base na origem do ataque (País, Rede e SO):

- Se ataque = “ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited” -> máquina atacante = windows, (88% de suporte e 67% de confiança).

Através de Apriori com dados obtidos do maior cluster resultante da aplicação do algoritmo K-Means para 3 *clusters*, com base em análise temporal do ataque (altura do dia e dia da semana):

- Se ataque = “ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited” e porta destino = comum -> servidor atacado = windows (70% de suporte e 60% de confiança).

Não foram obtidas novas regras através de Apriori com dados obtidos da aplicação de K-Means com base em análise de características do ataque (protocolo, nome do ataque, porto de origem e destino discretizados e número de acessos por sessão discretizado).

Através de Apriori com dados obtidos do maior cluster resultante da aplicação do algoritmo K-Means para 3 *clusters*, com base no alvo do ataque (sistema operativo do servidor e serviço que emula):

- Se servidor atacado = web -> máquina atacante = windows (75% de suporte e 62% de confiança).

O ataque “ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited”, segundo o sítio oficial do SNORT, é registado quando um datagrama deste tipo é detectado na rede, e pode ser lançado por variadas ferramentas ou *scripts*. Não é considerado uma grande ameaça.

As regras encontradas permitem a um administrador de rede/sistemas assumir que perante um ataque “ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited” detectado pelo sistema de detecção de intrusões, os servidores alvo serão *web*. O sistema de detecção de intrusões pode ser configurado para cortar o acesso do IP detectado como atacante a estes servidores.

Considerando que este ataque só é despoletado usando portas de origem entre 5 e 10 mil, caso estes portos não constem na análise ao tráfego normal de rede, pode-se configurar os sistemas de protecção para bloquearem tráfego iniciado nestas portas, caso permitam regras que incluam o porto de origem. Se os sistemas de protecção permitirem regras que contenham o número de acessos, pode

ser definida uma regra para bloquear tráfego originado num IP pertencente à rede 66.186.0.0/16 que tenha efectuado mais de 50 pedidos em poucas horas.

O conhecimento sobre o sistema operativo das máquinas atacantes revela que conjunto de aplicações é mais usada nos ataques. Isto permite que o administrador se possa informar sobre o seu funcionamento usar esse conhecimento para se proteger.

Perante um ataque "ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited" lançado através de acessos a portas comuns, os servidores atacados serão Windows. Face a esta regra, o sistema de detecção de intrusões pode ser configurado para cortar o acesso a estes servidores por parte do IP que é detectado como atacante e usa estas portas.

6.2.2 ICBAS - servidor externo (de 15/12/2009 a 17/05/2010)

Através de análise estatística (mera observação):

- Destacou-se o dia 01-01-2010 com aproximadamente 60% dos acessos, onde 99,9% dos servidores atacados são de email, e a maioria dos ataques ocorreu entre as 2h23m e as 2h35m através do IP 65.106.230.146 registado na Flórida(21h23 e 21h35 hora local), o mesmo identificado nos dados do ISEP. Foi retirado este conjunto dia e IP do resto da análise para evitar a sua influência nos resultados. Restaram para análise 17779 registos, dos 42865 iniciais.
- O protocolo mais utilizado foi TCP. UDP e ICMP foram pouco usados.
- 52% dos ataques foram dirigidos a sistemas Linux e 56% a servidores de email.
- Os portos mais atacados foram: 80, 110, 25, 443.
- Embora em apenas 58% dos acessos tenha sido possível registar o sistema operativo de origem, destacou-se Linux com 25% seguido por Windows com 17%.
- Nota-se que maior parte dos acessos ocorreu de madrugada.
- A maioria dos acessos foi, ordenado do mais frequente para o menos, proveniente dos EUA, Itália e China.
- Dos IPs de origem destacou-se um que participa em 18% dos acessos: 85.120.79.48 - ISP romeno (Internet Service Provider - Fornecedor de Serviços de Internet).
- O tipo de ataque identificado mais comum foi 'WEB-PHP Setup.php access'.
- Foram identificados poucos acessos considerados recorrentes. Conclui-se que ou os atacantes não repetiram o alvo ou tiveram sempre o cuidado de alterar o seu endereço IP.

Através do algoritmo Apriori:

- Filtrando os dados considerando apenas a rede “193.43.0.0/16”: se altura = madrugada -> servidor atacado = email (100% de suporte e 90% de confiança). Se servidor atacado = email -> altura = madrugada (90% de confiança e 100% de suporte).
- Filtrando os dados considerando apenas ataques a servidores de email: se porta destino = comum -> servidor atacado = linux (60% de confiança e 95% de suporte).
- Filtrando os dados considerando apenas acessos provenientes de um sistema operativo identificado como Windows: se servidor atacado = email -> rede = 85.120.0.0 (65% de confiança e 82% de suporte).
- Filtrando os dados considerando apenas acessos provenientes de um sistema operativo identificado como Windows: se servidor atacado = email -> número de acessos > 500 (65% de confiança e 82% de suporte).
- Filtrando os dados considerando apenas acessos registados como ataques do tipo “WEB-PHP Setup.php access”: se máquina atacante = linux -> país = Estados Unidos (61% de confiança e 79% de suporte).

Não foram obtidas novas regras através de Apriori com dados obtidos da aplicação de K-Means com base na origem do ataque (País, Rede e SO).

Através de Apriori com dados obtidos do maior cluster resultante da aplicação do algoritmo K-Means para 2 *clusters* com base em análise temporal do ataque (altura do dia e dia da semana):

- Se servidor atacado = email -> número de acessos > 500 (69% de confiança e 65% de suporte).

Não foram obtidos resultados através de Apriori com dados obtidos da aplicação de K-Means com base em análise de características do ataque (protocolo, nome do ataque, porto de origem e destino discretizados e número de acessos por sessão discretizado).

Através de Apriori com dados obtidos do maior *cluster* resultante da aplicação do algoritmo K-Means para 2 *clusters* com base no alvo do ataque (sistema operativo do servidor e serviço que emula):

- Se servidor atacado = linux -> servidor atacado = email (64% de confiança e 70% de suporte).

O ataque “WEB-PHP Setup.php access” é registado quando há uma tentativa de exploração de vulnerabilidades numa aplicação *web* PHP. Segundo o sítio oficial do SNORT, pode comprometer o sistema e permitir o acesso não autorizado ao sistema ou à aplicação. Como protecção apenas é sugerida a actualização do software e instalação de pacotes de correcções disponíveis.

Perante as regras anteriores pode-se assumir que os servidores de *email* correm mais perigo de ataque durante a madrugada. Como precaução pode-se restringir o acesso a estes servidores durante este período, dado que é uma altura em que poucos utilizadores fidedignos devem aceder. Através da análise dos *logs* da *firewall* e dos próprios servidores, consegue-se ter uma percepção de quais as redes que efectuam acessos de madrugada. Rejeitar os IPs destas redes, à excepção das portuguesas,

era uma medida que poderia trazer benefícios, dado que a grande maioria dos utilizadores que acede aos servidores de email encontra-se em Portugal. Outra possibilidade era, durante este período, apenas permitir acesso via *webmail* implementado num servidor *web*, bloqueando qualquer acesso directo ao servidor de *email* vindo do exterior da rede.

Depreende-se que os ataque aos servidores email que são dirigidos a portas vulgares serão dirigidos a servidores Linux. Estas portas habitualmente não são barradas pelas *firewalls*, o que implica que estes servidores fiquem expostos. Perante um elevado número de tentativas de ataque, pode-se estar perante sistemas comprometidos ou desactualizados e vulneráveis. Este conhecimento sugere uma análise dos serviços de *email* implementados em Linux relativamente à sua prevenção através de actualizações de *software*.

Os ataques a servidores de email foram relacionados com um número de acessos a cima dos 500. Perante esta regra pode-se configurar os sistemas de protecção para bloquearem acessos provenientes de acessos IP que já efectuaram mais de 500 pedidos dirigidos a um servidor de email em poucas horas.

Dado que se assume que os servidores Linux atacados são os que disponibilizam serviço de *email* pode-se libertar estes servidores de outros serviços que tenham e passá-los para outros servidores, podendo assim restringir mais o número de portas disponível para acesso.

6.2.3 FADEUP - servidor externo (de 26/12/2009 a 18/05/2010)

Através de análise estatística (mera observação):

- Registaram-se para análise 12701 registos.
- O protocolo mais utilizado foi ICMP. TCP também foi muito usado ao contrário de UDP.
- 45% dos ataques foram dirigidos a sistemas Linux e 55% a servidores de email.
- Os portos mais atacados foram: 80 e 110.
- Embora em apenas 8% dos acessos tenha sido possível registar o sistema operativo de origem, foram identificados sistemas Windows em 5% dos acessos e Linux em 3%.
- Nota-se que maior parte dos acessos ocorreu de tarde e a menor durante a noite. Contudo a distribuição de acessos por altura mostrou-se bastante equilibrada.
- A maioria dos acessos foi, ordenado do mais frequente para o menos, proveniente dos EUA, Portugal e China.
- Dos IPs de origem destacou-se um que participa em 8% dos acessos: 85.120.79.48 - ISP romeno (Internet Service Provider - Fornecedor de Serviços de Internet).
- O tipo de ataque identificado mais comum foi 'ICMP PING'.
- Foram identificados poucos acessos considerados recorrentes. Conclui-se que ou os atacantes não repetiram o alvo ou tiveram sempre o cuidado de alterar o seu endereço IP.

Através do algoritmo Apriori:

- Se protocolo = icmp -> ataque = ICMP PING (65% de confiança e 63% de suporte).
- Filtrando os dados considerando apenas a rede “85.120.0.0/16”: se servidor atacado = windows -> servidor atacado = web (99% de confiança e 74% de suporte). Se servidor atacado = web -> servidor atacado = windows (74% de confiança e 99% de suporte).
- Filtrando os dados considerando apenas a rede “85.120.0.0/16”: se altura = tarde -> número de acessos > 50 (100% de confiança e 99% de suporte). Se número de acessos > 50 -> altura = tarde (100% de confiança e 99% de suporte).
- Filtrando os dados considerando apenas ataques a servidores Windows: se porta destino = invulgar -> protocolo = icmp (100% de confiança e 70% de suporte).
- Filtrando os dados considerando apenas ataques a servidores Linux: se porto destino = invulgar -> ataque = ICMP PING (64% de confiança e 70% de suporte).
- Filtrando os dados considerando apenas acessos cujo porto destino é invulgar: se ataque = ICMP PING -> porto origem = não identificado (100% de confiança e 65% de suporte). Se porto origem = não identificado -> ataque = ICMP PING (65% de confiança e 100% de suporte).
- Filtrando os dados considerando apenas acessos efectuados por sistemas identificados como Windows: se rede = 85.120.0.0/16 -> número de acessos > 50 E altura = madrugada E servidor = windows E servidor = email E porta destino = comum (100% de confiança e 85% de suporte).
- Filtrando os dados considerando apenas acessos efectuados por sistemas identificados como Windows: se número de acessos > 50 E altura = madrugada -> servidor = windows E servidor = email E porto destino = comum (100% de confiança e 85% de suporte).
- Filtrando os dados considerando apenas acessos portugueses: se rede = 193.136.0.0/16 -> ataque = ICMP PING CyberKit 2.2 Windows (87% de confiança e 93% de suporte).
- Filtrando os dados considerando apenas acessos chineses: se altura = tarde -> servidor = windows (71% de confiança e 63% de suporte).

Não foram obtidas novas regras através de Apriori com dados obtidos da aplicação de K-Means com base na origem do ataque (País, Rede e SO).

Não foram obtidas novas regras através de Apriori com dados obtidos do maior cluster resultante da aplicação do algoritmo K-Means com base em análise temporal do ataque (altura do dia e dia da semana).

Através de Apriori com dados obtidos da aplicação de K-Means com base em análise de características do ataque (protocolo, nome do ataque, porto de origem e destino discretizados e número de acessos por sessão discretizado):

- Se servidor atacado = web -> porta destino = comum (100% de confiança e 65% de suporte).
- Se servidor atacado = windows -> porta destino = comum (100% de confiança e 64% de suporte).

Não foram obtidos resultados através de Apriori com dados obtidos do maior *cluster* resultante da aplicação do algoritmo K-Means para 2 *clusters* com base no alvo do ataque (sistema operativo do servidor e serviço que emula).

O ataque "ICMP PING" é registado quando é detectado um pedido *ping* tipicamente gerado pelo programa *nmap* (de tamanho zero). Segundo o sítio *web* oficial do SNORT, é usado para descobrir quais as portas em que um sistema aceita conexões. Como medida de reacção sugere-se o bloqueio de tráfego da máquina que efectua este tipo de pedidos, no caso de ela enviar tráfego suspeito após estes pedidos de reconhecimento.

O ataque "ICMP PING CyberKit 2.2 Windows" é registado quando é detectado um pedido ICMP echo de uma máquina Windows a executar o programa CyberKit 2.2. O seu perigo e reacção sugerida são equivalentes aos do ataque descrito anteriormente.

Dada a associação do protocolo ICMP a ataques de varredura de portas, e sabendo que estes ataques são habitualmente sucedidos por tentativas de intrusão, deve-se bloquear todo o tráfego que utilize este protocolo salvo excepções extremamente necessárias.

Conseguiu-se relacionar também a rede 85.120.0.0/16 com um número de acessos superior a cinquenta, hora pertencente ao intervalo definido como madrugada, servidor de email Windows e utilização de portas de destino comuns. Este conhecimento permite estabelecer regras nos mecanismos de segurança da instituição para bloquear tráfego que agregue estas características.

Das regras obtidas nota-se uma preferência dos atacantes em servidores web a correr num sistema operativo Windows. Perante esta ameaça é aconselhado verificar as configurações de segurança a nível de rede e dos próprios sistemas, tais como actualizações e outro software de protecção variado (*anti-vírus*, *anti-malware*, *firewall* local, etc.).

6.2.4 FCUP - servidor externo (de 11/12/2009 a 17/05/2010)

Através de análise estatística (mera observação):

- Destacou-se o dia 31-12-2009 com aproximadamente 60% dos acessos, onde 99,9% dos servidores atacados são de email, e os ataques ocorreram entre as 19h58m e as 20h03m através do IP 65.106.230.146 registado na Flórida(14h58 e 15h03 hora local), o mesmo identificado nos dados do ISEP e do ICBAS. Foi retirado este conjunto dia e IP do resto da análise para evitar a sua influência nos resultados. Restaram para análise 19400 registos, dos 47653 iniciais.
- O protocolo mais utilizado foi TCP. UDP e ICMP foram pouco usados.
- 49% dos ataques foram dirigidos a sistemas Linux e 57% a servidores de email.
- Os portos mais atacados foram: 80, 110, 25, 443 e 445.

- Embora em apenas 48% dos acessos tenha sido possível registar o sistema operativo de origem, destacou-se Linux com 25% seguido por Windows com 23%.
- Nota-se que maior parte dos acessos ocorreu de tarde.
- A maioria dos acessos foi, ordenado do mais frequente para o menos, proveniente dos EUA, China e Holanda.
- Dos IPs de origem destacaram-se dois que participam em 10% dos acessos cada um:
  - 85.120.79.48 - ISP romeno (Internet Service Provider - Fornecedor de Serviços de Internet).
  - 174.36.194.131 - Empresa de hosting americana
- O tipo de ataque identificado mais comum foi 'ICMP PING'.
- Foram identificados poucos acessos considerados recorrentes. Conclui-se que ou os atacantes não repetiram o alvo ou tiveram sempre o cuidado de alterar o seu endereço IP.

Através do algoritmo Apriori:

- Filtrando os dados considerando apenas a rede "10.0.0.0/16": se servidor = email -> servidor = windows (75% de suporte e 67% de confiança).
- Filtrando os dados considerando apenas a rede "10.0.0.0/16": se servidor = windows -> número de acessos > 500 (64% de confiança e 67% de suporte).
- Filtrando os dados considerando apenas a rede "10.0.0.0/16": se servidor = email -> número de acessos > 500 (64% de confiança e 67% de suporte).
- Filtrando os dados considerando apenas a rede "85.120.0.0/16": se altura = madrugada -> numero de acessos > 500 (91% de confiança e 61% de suporte).
- Filtrando os dados considerando apenas a rede "85.120.0.0/16": se servidor = windows -> porta de origem > 50000 e < 60000 (62% de confiança e 61% de suporte).
- Filtrando os dados considerando apenas acessos provenientes de um sistema operativo Linux: se servidor = email -> altura = tarde (64% de confiança e 64% de suporte).
- Filtrando os dados considerando apenas acessos a portas conhecidas: se rede = 10.0.0.0/16 -> servidor = email (66% de confiança e 99% de suporte).
- Filtrando os dados considerando apenas acessos a portas conhecidas: se rede = 10.0.0.0/16 -> servidor = windows (66% de confiança e 99% de suporte).
- Filtrando os dados considerando apenas acessos a portas conhecidas: se rede = 10.0.0.0/16 -> pais = Estados Unidos (94% de confiança e 99% de suporte).

- Filtrando os dados considerando apenas acessos provenientes da China: se número de acessos > 30 e < 500 -> servidor = web (99% de confiança e 70% de suporte).
- Filtrando os dados considerando apenas acessos provenientes da China: se altura = tarde -> servidor = web (95% de confiança e 61% de suporte).
- Filtrando os dados considerando apenas acessos provenientes da China: se número de acessos > 30 e < 500 -> rede = 218.90.0.0/16 (80% de confiança e 70% de suporte).
- Filtrando os dados considerando apenas acessos provenientes da China: se altura = tarde -> servidor = windows (77% de confiança e 61% de suporte).

Não foram obtidas novas regras através de Apriori com dados obtidos da aplicação de K-Means com base na origem do ataque (País, Rede e SO).

Através de Apriori com dados obtidos do maior cluster resultante da aplicação do algoritmo K-Means para 2 *clusters* com base em análise temporal do ataque (altura do dia e dia da semana):

- Se servidor = email -> servidor = linux (64% de confiança e 68% de suporte).

Não foram obtidas novas regras através de Apriori com dados obtidos da aplicação de K-Means com base em análise de características do ataque (protocolo, nome do ataque, porto de origem e destino discretizados e número de acessos por sessão discretizado).

Não foram obtidas novas regras através de Apriori com dados obtidos do maior *cluster* resultante da aplicação do algoritmo K-Means com base no alvo do ataque (sistema operativo do servidor e serviço que emula).

A partir das regras anteriores é possível definir um padrão de ataques correspondente a IPs da rede 10.0.0.0/16 que efectuem mais de 500 acessos, cortando-lhes o acesso a máquinas Windows servidoras de email.

Outro perfil possível são IPs da rede 85.120.0.0/16 com mais de 500 acessos de madrugada, sendo-lhes bloqueado o acesso a servidores Windows.

Pode ser ainda criado outro perfil para bloqueio a servidores web Windows, composto por IPs da rede 218.90.0.0/16 com mais de 30 e menos de 500 acessos durante a tarde.

6.2.5 FEP - servidor externo (de 18/01/2010 a 18/05/2010)

Através de análise estatística (mera observação):

- Registaram-se para análise 69711 registos.
- O protocolo mais utilizado foi TCP. ICMP e UDP não foram muito usados.
- 51% dos ataques foram dirigidos a sistemas Linux e 51,5% a servidores web.
- Os porto mais atacado foi: 445.

- Embora em apenas 46% dos acessos tenha sido possível registar o sistema operativo de origem, foram identificados sistemas Windows em 42% dos acessos e Linux em 4%.
- Nota-se que maior parte dos acessos ocorreu de tarde e a menor durante a noite. Contudo a distribuição de acessos por altura mostrou-se bastante equilibrada.
- A maioria dos acessos foi, ordenado do mais frequente para o menos, proveniente dos EUA, Portugal e China.
- Dos IPs de origem destacou-se um que participa em 6% dos acessos: 193.136.2.225 - videoconf03.fccn.pt.
- O tipo de ataque identificado mais comum foi 'ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited'.
- Como acessos considerados recorrentes salienta-se o intervalo de uma hora. Houve vários IPs a repetir o acesso passado uma hora do seu último contacto, perfazendo um total de 501 registos em que foi detectado este espaçamento temporal.

Através do algoritmo Apriori:

- Filtrando os dados considerando apenas a rede "193.136.0.0": se número de acessos > 40 -> ataque = 'ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited' (100% de confiança e 60% de suporte). Se ataque = 'ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited' -> número de acessos > 40 (85% de confiança e 71% de suporte).
- Filtrando os dados considerando apenas a rede "85.120.0.0/16": se número de acessos > 40 -> altura = tarde (90% de confiança e 100% de suporte).
- Filtrando os dados considerando apenas o protocolo TCP: se ataque = 'ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited' -> máquina atacante = windows (96% de confiança e 100% de suporte).
- Filtrando os dados considerando apenas ataques provenientes de máquinas Linux: se ataque = 'ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited' -> servidor = web (69% de confiança e 82% de suporte).
- Filtrando os dados considerando apenas ataques provenientes de Portugal: se rede = 193.136.0.0/16 e porta destino = invulgar -> ataque = 'ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited' (71% de confiança e 89% de suporte).
- Filtrando os dados considerando apenas ataques provenientes de Portugal: se rede = 193.136.0.0/16 -> número de acessos > 40 (60% de confiança e 89% de suporte).

Não foram obtidas novas regras através de Apriori com dados obtidos da aplicação de K-Means com base na origem do ataque (País, Rede e SO).

Não foram obtidas novas regras através de Apriori com dados obtidos do maior cluster resultante da aplicação do algoritmo K-Means com base em análise temporal do ataque (altura do dia e dia da semana).

Não foram obtidas novas regras através de Apriori com dados obtidos da aplicação de K-Means com base em análise de características do ataque (protocolo, nome do ataque, porto de origem e destino discretizados e número de acessos por sessão discretizado).

Através de Apriori com dados obtidos do maior *cluster* resultante da aplicação do algoritmo K-Means para 2 *clusters* com base no alvo do ataque (sistema operativo do servidor e serviço que emula).

- Se ataque = 'ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited' -> servidor = windows (63% de confiança e 73% de suporte).

Perante as regras anteriores pode-se considerar que, no caso da FEP, um número de acessos superior a quarenta num curto espaço de tempo será um ataque. Foi também associado a este elevado número de ocorrências o período correspondente à tarde. Desta forma pode-se criar regras nos mecanismos de segurança para bloquearem IPs de máquinas que façam tantos acessos num curto espaço de tempo, durante a tarde.

Os servidores web implementados em Windows foram considerados os mais propícios a serem atacados. Desta forma, sugere-se especial cuidado na segurança destas máquinas. Dado que o ataque mais comum registado foi dirigido a portas pouco comuns, é aconselhado abrir apenas as portas necessárias à disponibilização do serviço web. Com a aplicação destas sugestões prevê-se grandes melhorias a nível de segurança para este tipo de servidores.

6.2.6 Comparação

É feita na tabela 6.1 uma comparação entre os resultados obtidos, onde os campos "N" representam o número de registos recolhidos (N Análise refere-se apenas aos registos que não foram considerados anómalos ou impróprios), os campos "Windows, Linux, Web e Email" referem-se em percentagem aos sistemas atacados, o campo "Dias" representa o número de dias que o *honeypot* esteve em funcionamento, o campo "Dia invulgar" corresponde à eventualidade de ter havido um determinado dia com um número anormal de acessos, e os outros campos contêm os valores mais frequentes registados em cada faculdade.

Analisando todos os dados aglomerando-os como um todo, através de algoritmos de mineração de dados, apenas foi possível obter uma nova regra. Esta escassez de resultados pode dever-se à dispersão dos valores. Assim, mesmo aumentando a quantidade de dados, manteve-se o nível de conhecimento obtido. Usou-se o segundo maior *cluster* obtido através da execução do algoritmo K-Means para obtenção de três *clusters*, com base em características do ataque (protocolo, nome do ataque, portas de origem e destino discretizadas e número de acessos por sessão discretizado). A regra obtida foi:

Tabela 6.1: Comparativo dos resultados obtidos para cada faculdade

	ISEP	**ICBAS**	**FADEUP**	**FCUP**	**FEP**
N total	157635	42865	12701	47653	69711
N total/dia	1051	280	89	303	581
N analise	118664	17779	12701	19400	69711
N analise/dia	791	116	89	123	581
Dias	150	153	143	157	120
Dia invulgar	31/12/2009	01/01/2010		31/12/2009	
Protocolo	TCP	TCP	ICMP	TCP	TCP
Windows	57%	48%	55%	51%	51%
Linux	43%	52%	45%	49%	49%
Web	59%	44%	45%	43%	51%
EMail	41%	56%	55%	57%	49%
Portas	445,8335,135	80,110,25	80,110	80,110,25	445
Países	EUA,CN	EUA,IT	EUA,PT	EUA,CN	EUA,PT
Altura	tarde	madrugada	-	tarde	-
Ataque	ICMP...	WEB-PHP...	ICMP...	ICMP...	ICMP...
Intervalo	1,2,10	-	-	-	1

- Se servidor = web -> protocolo = udp (63% de confiança e 63% de suporte).

Em todos os casos foram detectados acessos nas portas habitualmente abertas e necessárias ao funcionamento dos serviços. Isto alerta para o perigo a que os servidores estão sujeitos mesmo que implementados atrás de uma *firewall*.

Nota-se que no primeiro e último dias do ano há maior probabilidade de ataque e, por isso, devem-se tomar medidas para prevenir estas prováveis tentativas de intrusão massivas. Os períodos de ataque registados diferem muito entre os vários casos de estudo, não sendo possível definir um período de perigo geral.

Por outro lado, o país de origem dos acessos é habitualmente Estados Unidos da América, Portugal ou China. Quanto a servidores alvo, não é possível identificar uma preferência, quer por serviço, quer por sistema operativo.

O protocolo mais usado é TCP, embora no caso da FADEUP se tenha registado uma maior utilização de ICMP. Quanto a acessos periódicos, apenas no caso do ISEP e da FEP se registou repetição. O intervalo de uma hora foi identificado nestes dois casos. No caso do ISEP foram ainda detectados intervalos de duas e dez horas.

Os ataques registados mais vezes não são considerados perigosos por si só, à excepção do “WEB-PHP Setup.php access” que pode comprometer o sistema. A maioria dos outros ataques tem como função o varrimento de portas e consequente identificação das portas abertas para comunicação. Servem como auxiliares para o lançamento de outros ataques. Por isto, aconselha-se o bloqueio de

tráfego ICMP vindo do exterior da rede, assim como a comunicação em portas que não são necessárias aos serviços disponibilizados.

Outra sugestão é o bloqueio de comunicações onde tenham sido efectuados mais de 50 acessos num curto espaço de tempo. Deve, no entanto, haver o cuidado de não englobar servidores onde se espera um número muito elevado de acessos por minuto, como um portal informativo por exemplo.

Por último, conseguiu-se detectar um padrão que surge em mais do que um caso, composto pela rede 85.120.0.0/16, mais de cinquenta acessos num dia, efectuados de madrugada e dirigidos a um servidor Windows.

Perante os resultados obtidos comprova-se que a aplicação de técnicas de mineração de dados à informação recolhida por *honeypots* proporciona um aumento do conhecimento obtido através da análise estatística. Este acréscimo pode ser muito útil a um administrador de sistemas e/ou rede devido à capacidade de definir regras e definir modelos capazes de aumentar os níveis de segurança, prevenir e reagir a ataques.

Segue-se a conclusão, onde consta um resumo do trabalho efectuado, crítica aos resultados obtidos e sugestões para trabalho futuro.

Capítulo 7

Conclusão

7.1 Objectivos realizados

Os objectivos delineados foram totalmente atingidos, nomeadamente:

- seleccionou-se um sistema de recolha de tentativas de intrusão constituído por *honeypots* de baixa interacção conjugado com IDS. Esta opção baseou-se no facto de terem um baixo risco de comprometimento, salvaguardando as redes das instituições colaboradoras. Por outro lado, os *honeypots* têm a capacidade de emular serviços Windows e Linux pelo que evitam a necessidade de aquisição de licenças. O *software* escolhido foi o Honeyd por ser o mais utilizado em investigação, muito poderoso e de código aberto. A este, juntou-se o IDS Snort em modo de detecção de ataques por forma a acrescentar ao *log* de acessos o nome do ataque lançado.

- a implementação do sistema foi efectuada em cinco faculdades portuguesas. Estas cederam a sua rede e forneceram computadores para a instalação da solução de recolha. Esta manteve-se em funcionamento durante vários meses, o que implicou a sua constante monitorização e extracção de *logs*, para salvaguarda da informação recolhida. A solução de recolha implementada comprovou ser muito eficaz. Conseguiu registar milhares de registos, tendo começado a ser acedida poucas horas após a sua implementação, e mantendo-se em operação durante cerca de meio ano.

- a ferramenta desenvolvida para fundir os *logs* recolhidos pelo Honeyd e Snort permite também eliminar dados incorrectos, derivar nova informação e uniformizar o resultado final num ficheiro CSV. Estes procedimentos permitiram à aplicação de mineração de dados usada (SPSS Clementine 12.0) utilizar os dados sem necessidade de nenhum ajuste adicional.

- seleccionou-se como técnicas de mineração, a serem aplicadas aos dados recolhidos, os algoritmos Apriori, K-Means e Redes Neuronais. Estes foram os algoritmos considerados mais usados nesta área e os mais indicados para o presente contexto, com base em artigos científicos de

áreas análogas. Não se conseguiu obter informação suficiente referente à abordagem do presente trabalho de mestrado, pelo que se teve de procurar outras áreas de utilização de mineração de dados em informação de *logs*. Por este motivo, entende-se que se contribui para a divulgação da utilização de mineração de dados e para a sua aplicação com um intuito pouco comum.

- a implementação dos algoritmos seleccionados mostrou-se eficaz, mas apenas para o caso do Apriori e K-Means, com os quais se conseguiu obter conhecimento capaz de aumentar os níveis de segurança das instituições colaboradoras. Este conhecimento permite relacionar características dos atacantes, relativamente ao seu país de origem e sistema operativo utilizado. Associa também detalhes dos seus métodos, tais como o nome do ataque utilizado, servidor atacado, protocolo e portas utilizadas. No caso particular da descoberta de conhecimento sobre a motivação do atacante, as técnicas utilizadas não são as mais adequadas, sendo esta temática de âmbito mais ligado à sociologia ou psicologia. Contudo, pode-se verificar estatisticamente que, no caso dos acessos ao ISEP, se destacaram atacantes da Faculdade de Engenharia do Porto (FEUP), cuja motivação pode ter por base a rivalidade entre estas faculdades. Por análise estatística evidencia-se também qual o desfasamento temporal, em número de horas, entre acessos do mesmo atacante. Através de Redes Neuronais não se conseguiu extrair conhecimento.

7.2 Trabalho futuro

Os resultados obtidos tiveram por base dados recolhidos de *honeypots* de baixa interactividade e, como tal, poderiam ser melhorados em quantidade e qualidade. Neste tipo de *honeypots* a ilusão criada ao atacante é facilmente desmascarada, pois o leque de respostas dadas ao atacante é limitado. Assim, deixa-se como sugestão para trabalho futuro a repetição desta experiência substituindo os *honeypots* de baixa interacção por outros de alta interacção ou mesmo *honeynets*.

Outra experiência interessante seria a comparação dos resultados obtidos com outros provenientes de um trabalho semelhante, que implementasse o sistema de recolha em instituições de um ramo diferente da educação como, por exemplo, em empresas.

O aumento de regras obtidas com algoritmos de mineração de dados estará directamente relacionado com um aumento de dados recolhidos. Desta forma, recolher informação durante mais tempo poderia ter vantagens. Por outro lado, poderia levar a resultados desactualizados. Como alternativa, poder-se-ia aumentar o número de *honeypots* implementados.

Bibliografia

Agrawal, R. and Srikant, R. (1994). Fast algorithm for mining association rules. *The international conference on very large data bases*.

Agrawal, R., Srikant, R., and Vu, Q. (1997). Mining association rules with item constraints. *The 3rd international conference on knowledge discovery in databases and data mining, Newport Beach, California*.

Alpaydin, E. (2004). *Introduction To Machine Learning*. Massachussetts Institute of Technology.

Artail, H., Safa, H., Sraj, M., Kuwatly, I., and Al-Masri, Z. (2006). A hybrid honeypot framework for improving intrusion detection systems in protecting organizational networks. *Computers and Security*.

Aydin, M. A., Zaim, A. H., and Ceylan, K. G. (2009). A hybrid intrusion detection system design for computer network security. *Computers and Electrical Engineering*.

Baecher, P., Koetter, M., Holz, T., Dornseif, M., and Freiling, F. (2006). The nepenthes platform: an efficient approach to collect malware. *Proceedings of the 9th international symposium on recent advances in intrusion detection (RAID)*.

Bailey, M. and Cooke, E. (2004). Tracking global threats with the internet motion sensor. *Nanog 32*.

Bailey, M., Cooke, E., Jahanian, F., Myrick, A., and Sinha, S. (2006). Practical darknet measurement. *40th Annual Conference on Information Sciences and Systems*.

Bailey, M., Cooke, E., Jahanian, F., Nazario, J., and Watson, D. (2005). The internet motion sensor: A distributed blackhole monitoring system. *NDSS*.

Baldissera, T. A. and Nunes, R. C. (2007). Impacto na implementação da norma nbr iso/iec 17799 para a gestão da segurança da informação em colégios: um caso de estudo. *XXVII Encontro Nacional de Engenharia de Produção*.

Beheshti, M., Han, J., Kowalski, K., Ortiz, J., Tomelden, J., and Alvillar, D. (2008). Packet information collection and transformation for network intrusion detection and prevention. *Internatioal Symposium on Telecommunications*.

Bradley, T. (2010). Nsa "perfect citizen"raises "big brother"concerns in the private sector. *PC World*.

Brugger, S. (2004). Data mining for network intrusion detection.

Carsten (2007). Definitions of the terms hacker, cracker and coder. *http://www.roysac.com/blog/2007/09/definitions-of-terms-hacker-cracker-and.html*.

Carvalho, N. (2009). Organizações e segurança informática.

Center, I. S. (2010). About the internet storm center.

Chen, Z., Gao, L., and Kwiat, K. (2003). Modeling the spread of active worms.

Chuvakin, D. A. (2005). Log data mining. *http://www.chuvakin.org/*.

Collins, M. P., Shimeall, T. J., Faber, S., Janies, J., Weaver, R., Shon, M. D., and et al. (2007). Using uncleanliness to predict future botnet addresses. *IMC '07: Proceedings of the seventh ACM SIGCOMM conference on Internet measurement*.

Computer-Security.Org.UK (2007). Computer security threats.

Cooke, E., Bailey, M., Mao, Z. M., Watson, D., Jahanian, F., and McPherson, D. (2004). Toward understanding distributed blackhole placement. *Conference on Computer and Communications Security*.

Cyber-TA (2010). About cyber-ta. *http://www.cyber-ta.org/*.

Cymru, T. (2010). http://www.team-cymru.org/.

Dagon, D., Zou, C., and Lee, W. (2006). Modeling botnet propagation using time zones. *Proceedings of the 13th annual network and distributed system security symposium (NDSS'06)*.

Fielding, J. (2007). Create a simple honeypot with debian and nepenthes. *TechRepublic*.

Frawley, W. J., Shapiro, G. P., and Matheus, C. J. (1991). Knowledge discovery in databases: An overview. *The AAAI workshop on knowledge discovery in databases*.

Freisleben, B. and Galeas, P. (2009). Semantic neighborhood analysis. *http://www.galeas.de/phd.html*.

Gagadis, F. and Wolthusen, S. D. (2008). Topological models and effectiveness of network telescopes. *TechTarget*.

Gaudin, S. (2002). Honeypots turn the tables on hackers. *Datamation*, 1:1.

Giudic, P. and Castelo, R. (2001). Association models for web mining. *Kluwer Academic Publishers*.

Google (2010). Porque usar o google. *Página oficial do Google*.

Grégio, A., Santos, R., and Montes, A. (2007). Evaluation of data mining techniques for suspicious network activity classification using honeypots data. *Data Mining, Intrusion Detection, Information Assurance, and Data Networks Security*.

Han, J. and Fu, Y. (1995). Discovery of multiple-level association rules from large database. *The 21st international conference on very large data bases, Zurich, Switzerland*.

Harrop, W. and Armitage, G. (2005). Defining and evaluating greynets (sparse darknets). *IEEE Computer Society*.

Hess, K. (2009). Tempting those blackhatted poohs with a taste of virtual honey. *Linux Magazine*.

Hoepers, C., Steding-Jessen, K., and Chaves, M. (2007). Honeypots e honeynets: Definições e aplicações. *http://www.cert.br/docs/whitepapers/honeypots-honeynets/*.

Hong, T.-P., Wud, Y.-Y., and Wang, S.-L. (2009). An effective mining approach for up-to-date patterns. *Expert Systems with Applications*.

Inc., T. (2010). Stop the bots and block the trojans. *http://www.threatstop.com*.

Khribi, M. K., Jemni, M., and Nasraoui, O. (2009). Automatic recommendations for e-learning personalization based on web usage mining techniques and information retrieval. *Educational Technology & Society*.

Li, D. and Deogun, J. S. (2005). Discovering partial periodic sequential association rules with time lag in multiple sequences for prediction. *Lecture Notes in Computer Sciences*.

Li, X. and Chandra, C. (2007). A knowledge integration framework for complex network management. *Industrial Management & Data Systems*.

Li, Y., Ning, P., Wang, X. S., and Jajodia, S. (2003). Discovering calendar-based temporal association rules. *Data and Knowledge Engineering*.

Mannila, H., Toivonen, H., and Verkamo, I. (1994). Efficient algorithm for discovering association rules. *The AAAI workshop knowledge discovery in databases*.

Mikhalenko, P. (2006). Managing a honeypot. *O'Reilly Sys Admin*.

Moore, D., Shannon, C., Voelkery, G. M., and Savagey, S. (2004). Network telescopes: Technical report. *CAIDA - Cooprerative Association for Internet Data Analysis*.

Morais, J. (2007). Web mining, data mining, etc... *http://ante-et-post.weblog.com.pt/2007/06/web_mining_data_mining_etc*.

netVigilance (2010). http://www.netvigilance.com.

O'Neill, L. T. (2007). Ids vs. ips explained. *Network Security Journal*.

Ozden, B., Ramaswamy, S., and Silberschatz, A. (1998). Cyclic association rules. *The 14th international conference on data engineering, Orlando, Florida, USA*.

Paarl (2010). The impact of page rank. *HubPages*.

Park, J. S., Chen, M. S., and Yu, P. S. (1997). Using a hash-based method with transaction trimming for mining association rules. *IEEE Transactions on Knowledge and Data Engineering*.

PhishTank (2010). What is pishtank. *http://www.phishtank.com/*.

Pouget, F. and Dacier, M. (2003). Honeypot, honeynet: A comparative survey. *http://www.eurecom.fr/util/publidownload.en.htm?id=1273*.

Project, H. (2006). Know your enemy: Honeynets. *http://old.honeynet.org/papers/honeynet/*.

Provos, N. (2002). Honeyd manual.

Provos, N. (2004). A virtual honeypot framework. *Proceedings of the 13th USENIX security symposium*.

Raynal, F., Berthier, Y., Biondi, P., and Kaminsky, D. (2004). Honeypot forensics, part ii: analyzing the compromised host. *Security & Privacy, IEEE*.

Riordan, J., Zamboni, D., and Duponchel, Y. (2006). Building and deploying billy goat, a worm-detection system. *Proceedings of the 18th annual FIRST conference*.

Rist, L. (2009). Glastopf project. *The Honeynet Project*.

Roddick, J. F. and Spiliopoulou, M. (2002). A survey of temporal knowledge discovery paradigms and methods. *IEEE Transactions on Knowledge and Data Engineering*.

Saúde, S. (2010). Hospital de braga implementa plataforma de informação “paper-free”. *Sapo - Saúde*.

Sardana, A. and Joshi, R. (2009). An auto-responsive honeypot architecture for dynamic resource allocation and qos adaptation in ddos attacked networks. *Computer Communications*.

Savasere, A., Omiecinski, E., and Navathe, S. (1995). An efficient algorithm for mining association rules in large databases. *The ACM VLDB conference*.

Schenker, A., Bunke, H., Last, M., and Kandel, A. (2005). *Graph-Theoretic Techniques for Web Content Mining*. World Scientific.

Searchers, B. (2010). Internet status. *http://www.bestsearchers.com*.

Security, H. (2010). Cyber storm: Securing cyber space. *Homeland Security*.

Seifried, K. (2002). Honeypotting with vmware - basics. *http://seifried.org*, 1:1.

Shadowserver (2010). Mission; network based detection strategies. *http://www.shadowserver.org*.

Singhal, A. and Jajodia, S. (2006). Data warehousing and data mining techniques for intrusion detection systems. *Distrib Parallel Databases*.

SourceForge (2010). http://sourceforge.net/.

Spitzner, L. (1999). To build a honeypot. *http://www.spitzner.net*, 1:1.

Spitzner, L. (2003a). Definitions and value of honeypots. *http://www.tracking-hackers.com*, 1:1.

Spitzner, L. (2003b). Honeypot solutions - so you want to build your own honeypot. *http://www.tracking-hackers.com*, 1:1.

Spitzner, L. (2003c). Honeypots: Simple, cost-effective detection. *Security Focus TM*.

Spitzner, L. (2003d). Honeytokens: The other honeypot. *Security Focus TM*.

Spitzner, L. (2003e). Open source honeypots: Learning with honeyd. *Security Focus TM*, 1:1.

Srikant, R. and Agrawal, R. (1995). Mining generalized association rules. *The 21st international conference on very large data bases, Zurich, Switzerland*.

Stats, I. W. (2009). Internet usage statistics. *http://www.internetworldstats.com/stats.htm*.

Steding-Jessen, K. and Chaves, M. (2008). Implantacão de honeypots de baixa interatividade com honeyd e nepenthes. *Campus Party Brasil*.

Su, M.-Y. (2009). Discovery and prevention of attack episodes by frequent episodes mining and finite state machines. *Journal of Network and Computer Applications*.

Thonnard, O. and Dacier, M. (2008). A framework for attack patterns' discovery in honeynet data. *Elsevier*.

Vaarandi, R. (2004). A breadth-first algorithm for mining frequent patterns from event logs. *IFIP - International Federation for Information Processing*.

Veerasamy, N., Mokhomoana, P., and Vorster, J. (2009). Applying data-mining techniques in honeypot analysis. *Tech Republic*.

Watson, D. (2007). Honeynets: a tool for counterintelligence in online security. *Network Security*.

X., L., Bian, F., Zhang, H., Diot, C., Govindan, R., Hong, W., and et al (2005). Advanced indexing techniques for wide-area network monitoring. *First IEEE international workshop on networking meets databases (NetDB)*.

Yeldi, S. and et al. (2003). Enhancing network intrusion detection system with honeypot. *Pune Institute of Computer Technology*.

Zhang, J., Porras, P., and Ullrich, J. (2008a). Highly predictive blacklisting. *Usenix*.

Zhang, J., Zulkernine, M., and Haque, A. (2008b). Random-forests-based network intrusion detection systems. *IEEE Transactions on Systems, Man, and Cybernetics*.

Glossário

Autorooter	Programa desenhado para farejar a rede à procura de máquinas fragilizadas e as atacar. Dão ao atacante o controlo de uma máquina remota, sem esforço e num curto espaço de tempo.
Blackhat	Alguém com elevados conhecimentos em redes, sistemas e programação e que usa esse conhecimento para entrar em sistemas de forma ilícita para efectuar actos de vandalismo informático ou outras actividades ilegais.
Botnet	*Robot* por software ou *bots* que são executados automaticamente e sem a percepção do utilizador. A sua função é recolher informação sobre todas as acções do utilizador.
Carding	Termo usado para um movimento efectuado um cartão de crédito roubado por via electrónica para verificar se este ainda se encontra activo. Normalmente envolve quantias muito baixas para não ultrapassar o *plafond* e não levantar suspeitas junto do seu emissor.
DDoS Extortion Rackets	Acto de ameaçar entidades com o bloqueio dos seus sistemas através de ataques distribuídos de negação de serviço (*Distributed Denial of Service*).
Farming	Técnica de obter informações sobre utilizadores. Baseia-se na cópia de sítios web de instituições credíveis para levar o visitante a introduzir as suas credenciais ou outros dados privados. Recorre ao comprometimento de servidores DNS para que quando os alvos solicitam uma resolução de nome de um determinado endereço web sejam encaminhados para um site forjado em vez do site verdadeiro.
Fhishing	Técnica de obter informações sobre utilizadores. Baseia-se na cópia de sítios web de instituições credíveis para levar o visitante a introduzir as suas credenciais ou outros dados privados. Recorre ao envio de email para chegar aos alvos, onde o atacante se faz passar por uma instituição credível.

Flash worm	*Worm* que conhece previamente um conjunto de IPs de máquinas vulneráveis e através dessa lista consegue-se espalhar muito rapidamente.
Malware	MALicious softWARE, programas destinados a se infiltrarem em sistemas de forma ilícita e executar operações que causem dano ou recolha de informações.
Máquinas *zombie*	Sistemas infectados que podem ser usados por terceiros sem conhecimento ou autorização do utilizador.
OSI	*Open Systems Interconnection* - arquitectura que tem por objectivo definir um padrão de comunicação para possibilitar interacção entre dispositivos de fabricantes diferentes. Descreve um modelo composto por sete camadas. Os dispositivos podem não as implementar na totalidade mas têm obrigatoriamente de implementar todas as que se encontram a baixo da camada máxima que pretendem suportar.
Proxy honeypot	*Honeypot* que implementa/emula o serviço de *proxy* aberto. Este serviço funciona como um intermediário entre a máquina cliente e o destino e é usado para esconder a origem do tráfego. Por isso é muito usado em ataques e para fugir a restrições de tráfego nacional.
Relay honeypot	*Honeypot* que implementa/emula o serviço de *relay* aberto. Este serviço é basicamente um servidor de envio de email (SMTP) que não requer autenticação nem faz nenhum outro controlo de acesso. Por isso é muito usado para envio de SPAM.
SMB	*Server Message Block* ou *Common Internet File System* (CIFS) - protocolo usado para partilha de ficheiros e impressoras, principalmente em redes Windows.
Sticky honeypot	Também conhecido por *tarpit* tem a particularidade de responder de forma a tentar bloquear as máquinas dos atacantes sempre que é acedido.
Theft	Técnicas em que o atacante se faz passar pela vítima sem o seu consentimento.
Wrapper	Programa que executa ou “embrulha” (*wrap*) um conjunto de operações.
Worm	Verme, em português, à semelhança dos vírus é software que tem por objectivo tomar acções maliciosas e auto-replicar-se. Difere dos vírus porque estes precisam de um programa hospedeiro para se executar enquanto que os *worms* são programas completos e independentes.

Anexos

Apêndice A

Resultados obtidos com o SPSS Clementine 12.0

Resultados da aplicação do algoritmo Apriori aos dados recolhidos no ICBAS

Consequent	Antecedent	Support %	Confidence %
protocolo = tcp	porto_destino_perigoso = conhecido	93,987	100,000
protocolo = tcp	ataque = and porto_destino_perigoso = conhecido	68,755	100,000
protocolo = tcp	ataque =	74,419	98,133
porto_destino_perigoso = conhecido	protocolo = tcp	98,262	95,650
porto_destino_perigoso = conhecido	ataque = and protocolo = tcp	73,030	94,147
porto_destino_perigoso = conhecido	ataque =	74,419	92,389
ataque =	protocolo = tcp	98,262	74,322
ataque =	porto_destino_perigoso = conhecido	93,987	73,154
ataque =	porto_destino_perigoso = conhecido and protocolo = tcp	93,987	73,154
SO_Origem_Disc = vazio	ataque =	74,419	60,290
SO_Origem_Disc = vazio	ataque = and porto_destino_perigoso = conhecido	68,755	60,209
SO_Origem_Disc = vazio	ataque = and porto_destino_perigoso = conhecido and protocolo = tcp	68,755	60,209

Resultados da aplicação do algoritmo Apriori aos dados recolhidos no ICBAS, com IP da rede 85.120.0.0/16

Consequent	Antecedent	Support %	Confidence %
pais = ERRO-BD	ataque = ERRO-BD	73,031	99,750
ataque = ERRO-BD	pais = ERRO-BD	73,092	99,667

Resultados da aplicação do algoritmo Apriori aos dados recolhidos no ICBAS, com IP da rede 193.43.0.0/16

Consequent	Antecedent	Support %	Confidence %
servidor = mail	altura = madrugada	89,330	100,000
altura = madrugada	servidor = mail	99,938	89,385

Resultados da aplicação do algoritmo Apriori aos dados recolhidos no ICBAS, com protocolo TCP

Consequent	Antecedent	Support %	Confidence %
porto_destino_perigoso = conhecido	ataque =	74,322	94,147
ataque =	porto_destino_perigoso = conhecido	95,650	73,154
SO_Origem_Disc = vazio	ataque = and porto_destino_perigoso = conhecido	69,971	60,209

Resultados da aplicação do algoritmo Apriori aos dados recolhidos no ICBAS, com sistema atacado Windows

Consequent	Antecedent	Support %	Confidence %
protocolo = tcp	porto_destino_perigoso = conhecido	93,584	100,000
protocolo = tcp	ataque = and porto_destino_perigoso = conhecido	64,700	100,000
protocolo = tcp	ataque =	70,799	97,793
protocolo = tcp	SO_Origem_Disc = vazio	61,022	97,420
porto_destino_perigoso = conhecido	protocolo = tcp	98,120	95,377
porto_destino_perigoso = conhecido	ataque = and protocolo = tcp	69,236	93,449
porto_destino_perigoso = conhecido	SO_Origem_Disc = vazio	61,022	93,414
porto_destino_perigoso = conhecido	ataque =	70,799	91,386
ataque =	SO_Origem_Disc = vazio	61,022	76,449
ataque =	protocolo = tcp	98,120	70,563
ataque =	porto_destino_perigoso = conhecido	93,584	69,136
ataque =	porto_destino_perigoso = conhecido and protocolo = tcp	93,584	69,136
SO_Origem_Disc = vazio	ataque = and porto_destino_perigoso = conhecido	64,700	66,346
SO_Origem_Disc = vazio	ataque = and porto_destino_perigoso = conhecido and protocolo = tcp	64,700	66,346
SO_Origem_Disc = vazio	ataque =	70,799	65,892
SO_Origem_Disc = vazio	ataque = and protocolo = tcp	69,236	65,530
SO_Origem_Disc = vazio	porto_destino_perigoso = conhecido	93,584	60,912
SO_Origem_Disc = vazio	porto_destino_perigoso = conhecido and protocolo = tcp	93,584	60,912
SO_Origem_Disc = vazio	protocolo = tcp	98,120	60,587

Resultados da aplicação do algoritmo Apriori aos dados recolhidos no ICBAS, com sistema atacado Linux

Consequent	Antecedent	Support %	Confidence %
protocolo = tcp	porto_destino_perigoso = conhecido	94,358	100,000
protocolo = tcp	servidor = mail and porto_destino_perigoso = conhecido	61,485	100,000
protocolo = tcp	ataque = and porto_destino_perigoso = conhecido	72,478	100,000
protocolo = tcp	servidor = mail	64,290	98,641
protocolo = tcp	ataque =	77,743	98,418
porto_destino_perigoso = conhecido	servidor = mail and protocolo = tcp	63,416	96,955
porto_destino_perigoso = conhecido	protocolo = tcp	98,392	95,899
porto_destino_perigoso = conhecido	servidor = mail	64,290	95,637
porto_destino_perigoso = conhecido	ataque = and protocolo = tcp	76,513	94,726
porto_destino_perigoso = conhecido	ataque =	77,743	93,228
ataque =	protocolo = tcp	98,392	77,763
ataque =	porto_destino_perigoso = conhecido	94,358	76,812
ataque =	porto_destino_perigoso = conhecido and protocolo = tcp	94,358	76,812
ataque =	servidor = mail	64,290	73,955
ataque =	servidor = mail and protocolo = tcp	63,416	73,886
ataque =	servidor = mail and porto_destino_perigoso = conhecido	61,485	73,065
ataque =	servidor = mail and porto_destino_perigoso = conhecido and protocolo = tcp	61,485	73,065
servidor = mail	porto_destino_perigoso = conhecido	94,358	65,161
servidor = mail	porto_destino_perigoso = conhecido and protocolo = tcp	94,358	65,161
servidor = mail	protocolo = tcp	98,392	64,452
servidor = mail	ataque = and porto_destino_perigoso = conhecido	72,478	61,983
servidor = mail	ataque = and porto_destino_perigoso = conhecido and protocolo = tcp	72,478	61,983
altura = madrugada	servidor = mail and porto_destino_perigoso = conhecido	61,485	61,309
altura = madrugada	servidor = mail and porto_destino_perigoso = conhecido and protocolo = tcp	61,485	61,309
servidor = mail	ataque = and protocolo = tcp	76,513	61,238
servidor = mail	ataque =	77,743	61,157

Resultados da aplicação do algoritmo Apriori aos dados recolhidos no ICBAS, com sistema atacado Web

Consequent	Antecedent	Support %	Confidence %
protocolo = tcp	porto_destino_perigoso = conhecido	93,272	100,000
protocolo = tcp	ataque = and porto_destino_perigoso = conhecido	74,407	100,000
protocolo = tcp	ataque =	80,725	98,238
porto_destino_perigoso = conhecido	protocolo = tcp	98,167	95,013
porto_destino_perigoso = conhecido	ataque = and protocolo = tcp	79,303	93,827
porto_destino_perigoso = conhecido	ataque =	80,725	92,173
ataque =	protocolo = tcp	98,167	80,783
ataque =	porto_destino_perigoso = conhecido	93,272	79,775
ataque =	porto_destino_perigoso = conhecido and protocolo = tcp	93,272	79,775
SO_Origem_Disc = vazio	ataque = and porto_destino_perigoso = conhecido	74,407	62,160
SO_Origem_Disc = vazio	ataque = and porto_destino_perigoso = conhecido and protocolo = tcp	74,407	62,160
SO_Origem_Disc = vazio	ataque =	80,725	62,057
SO_Origem_Disc = vazio	ataque = and protocolo = tcp	79,303	61,652

Resultados da aplicação do algoritmo Apriori aos dados recolhidos no ICBAS, com sistema atacado Email

Consequent	Antecedent	Support %	Confidence %
protocolo = tcp	porto_destino_perigoso = conhecido	94,547	100,000
protocolo = tcp	ataque = and porto_destino_perigoso = conhecido	64,334	100,000
protocolo = tcp	ataque =	69,487	98,038
porto_destino_perigoso = conhecido	protocolo = tcp	98,336	96,147
porto_destino_perigoso = conhecido	ataque = and protocolo = tcp	68,123	94,438
porto_destino_perigoso = conhecido	ataque =	69,487	92,585
ataque =	protocolo = tcp	98,336	69,276
ataque =	porto_destino_perigoso = conhecido	94,547	68,045
ataque =	porto_destino_perigoso = conhecido and protocolo = tcp	94,547	68,045
so_destino = linux	ataque = and porto_destino_perigoso = conhecido	64,334	64,880
so_destino = linux	ataque = and porto_destino_perigoso = conhecido and protocolo = tcp	64,334	64,880
so_destino = linux	ataque = and protocolo = tcp	68,123	63,905
so_destino = linux	ataque =	69,487	63,575
altura = madrugada	ataque = and porto_destino_perigoso = conhecido	64,334	61,624
altura = madrugada	ataque = and porto_destino_perigoso = conhecido and protocolo = tcp	64,334	61,624
so_destino = linux	porto_destino_perigoso = conhecido	94,547	60,422
so_destino = linux	porto_destino_perigoso = conhecido and protocolo = tcp	94,547	60,422
RecordCount_perigoso = >=500	porto_destino_perigoso = conhecido	94,547	60,115
RecordCount_perigoso = >=500	porto_destino_perigoso = conhecido and protocolo = tcp	94,547	60,115

Resultados da aplicação do algoritmo Apriori aos dados recolhidos no ICBAS, com portas comuns

Consequent	Antecedent	Support %	Confidence %
SO_Origem_Disc = vazio	ataque =	73,154	60,209

Resultados da aplicação do algoritmo Apriori aos dados recolhidos no ICBAS, com portas invulgares

Consequent	Antecedent	Support %	Confidence %
ataque =	protocolo = tcp	71,094	100,000
ataque =	SO_Origem_Disc = vazio	62,956	91,679
protocolo = tcp	ataque =	94,200	75,472
SO_Origem_Disc = vazio	ataque =	94,200	61,271
protocolo = tcp	SO_Origem_Disc = vazio	62,956	60,624

Resultados da aplicação do algoritmo Apriori aos dados recolhidos no ICBAS, com país de origem EUA

Consequent	Antecedent	Support %	Confidence %
protocolo = tcp	porto_destino_perigoso = conhecido	98,063	100,000
protocolo = tcp	SO_Origem_Disc = vazio and porto_destino_perigoso = conhecido	63,863	100,000
protocolo = tcp	ataque = and porto_destino_perigoso = conhecido	90,953	100,000
protocolo = tcp	SO_Origem_Disc = vazio and ataque = and porto_destino_perigoso = conhecido	63,704	100,000
ataque =	SO_Origem_Disc = vazio and protocolo = tcp	64,261	99,752
ataque =	SO_Origem_Disc = vazio and porto_destino_perigoso = conhecido	63,863	99,751
ataque =	SO_Origem_Disc = vazio and porto_destino_perigoso = conhecido and protocolo = tcp	63,863	99,751
protocolo = tcp	ataque =	92,598	99,542
porto_destino_perigoso = conhecido	SO_Origem_Disc = vazio and protocolo = tcp	64,261	99,381
porto_destino_perigoso = conhecido	SO_Origem_Disc = vazio and ataque = and protocolo = tcp	64,102	99,379
protocolo = tcp	SO_Origem_Disc = vazio and ataque =	64,526	99,342
ataque =	SO_Origem_Disc = vazio	64,977	99,306
protocolo = tcp	SO_Origem_Disc = vazio	64,977	98,898
porto_destino_perigoso = conhecido	protocolo = tcp	99,284	98,771
porto_destino_perigoso = conhecido	SO_Origem_Disc = vazio and ataque =	64,526	98,725
porto_destino_perigoso = conhecido	ataque = and protocolo = tcp	92,173	98,676
porto_destino_perigoso = conhecido	SO_Origem_Disc = vazio	64,977	98,285
porto_destino_perigoso = conhecido	ataque =	92,598	98,223
ataque =	protocolo = tcp	99,284	92,838
ataque =	porto_destino_perigoso = conhecido	98,063	92,749
ataque =	porto_destino_perigoso = conhecido and protocolo = tcp	98,063	92,749
SO_Origem_Disc = vazio	ataque = and porto_destino_perigoso = conhecido	90,953	70,041
SO_Origem_Disc = vazio	ataque = and porto_destino_perigoso = conhecido and protocolo = tcp	90,953	70,041
SO_Origem_Disc = vazio	ataque =	92,598	69,685
SO_Origem_Disc = vazio	ataque = and protocolo = tcp	92,173	69,545
so_destino = windows	SO_Origem_Disc = vazio and porto_destino_perigoso = conhecido	63,863	65,933
so_destino = windows	SO_Origem_Disc = vazio and porto_destino_perigoso = conhecido and protocolo = tcp	63,863	65,933
so_destino = windows	SO_Origem_Disc = vazio and protocolo = tcp	64,261	65,896
so_destino = windows	SO_Origem_Disc = vazio and ataque = and porto_destino_perigoso = conhecido	63,704	65,889
so_destino = windows	SO_Origem_Disc = vazio and ataque = and protocolo = tcp	64,102	65,853
so_destino = windows	SO_Origem_Disc = vazio and ataque =	64,526	65,748
so_destino = windows	SO_Origem_Disc = vazio	64,977	65,741
SO_Origem_Disc = vazio	porto_destino_perigoso = conhecido	98,063	65,124
SO_Origem_Disc = vazio	porto_destino_perigoso = conhecido and protocolo = tcp	98,063	65,124
SO_Origem_Disc = vazio	protocolo = tcp	99,284	64,725
so_destino = windows	porto_destino_perigoso = conhecido	98,063	60,011
so_destino = windows	porto_destino_perigoso = conhecido and protocolo = tcp	98,063	60,011

Resultados da aplicação do algoritmo Apriori aos dados recolhidos no ICBAS, com ataque “WEB-PHP Setup.php access”

Consequent	Antecedent	Support %	Confidence %
SO_Origem_Disc = Linux	so_destino = windows	82,863	95,550
so_destino = windows	SO_Origem_Disc = Linux	96,312	82,207
pais = UNITED STATES	so_destino = windows and SO_Origem_Disc = Linux	79,176	60,822

Resultados da aplicação do algoritmo K-Means (para 3 clusters com os parâmetros país de origem, sistema operativo de origem e sub-rede) e Apriori aos dados recolhidos no ICBAS

Consequent	Antecedent	Support %	Confidence %
protocolo = tcp	porto_destino_perigoso = conhecido	93,491	100,000
protocolo = tcp	ataque = and porto_destino_perigoso = conhecido	71,180	100,000
protocolo = tcp	ataque =	77,147	97,380
porto_destino_perigoso = conhecido	protocolo = tcp	97,437	95,950
porto_destino_perigoso = conhecido	ataque = and protocolo = tcp	75,126	94,748
porto_destino_perigoso = conhecido	ataque =	77,147	92,265
ataque =	protocolo = tcp	97,437	77,102
ataque =	porto_destino_perigoso = conhecido	93,491	76,135
ataque =	porto_destino_perigoso = conhecido and protocolo = tcp	93,491	76,135

Resultados da aplicação do algoritmo K-Means (para 2 clusters com os parâmetros altura do dia e dia da semana) e Apriori aos dados recolhidos no ICBAS

Consequent	Antecedent	Support %	Confidence %
protocolo = tcp	porto_destino_perigoso = conhecido	94,484	100,000
protocolo = tcp	servidor = mail and porto_destino_perigoso = conhecido	62,288	100,000
protocolo = tcp	ataque = and porto_destino_perigoso = conhecido	65,692	100,000
protocolo = tcp	altura = madrugada	60,200	99,028
protocolo = tcp	servidor = mail	65,050	98,764
protocolo = tcp	ataque =	70,875	98,315
porto_destino_perigoso = conhecido	altura = madrugada	60,200	97,072
porto_destino_perigoso = conhecido	servidor = mail and protocolo = tcp	64,246	96,952
porto_destino_perigoso = conhecido	protocolo = tcp	98,473	95,949
porto_destino_perigoso = conhecido	servidor = mail	65,050	95,754
porto_destino_perigoso = conhecido	ataque = and protocolo = tcp	69,681	94,275
porto_destino_perigoso = conhecido	ataque =	70,875	92,687
ataque =	altura = madrugada	60,200	75,533
servidor = mail	altura = madrugada	60,200	73,927
RecordCount_perigoso = >=500	servidor = mail and porto_destino_perigoso = conhecido	62,288	71,749
RecordCount_perigoso = >=500	servidor = mail and porto_destino_perigoso = conhecido and protocolo = tcp	62,288	71,749
ataque =	protocolo = tcp	98,473	70,761
altura = madrugada	servidor = mail and porto_destino_perigoso = conhecido	62,288	70,249
altura = madrugada	servidor = mail and porto_destino_perigoso = conhecido and protocolo = tcp	62,288	70,249
RecordCount_perigoso = >=500	servidor = mail and protocolo = tcp	64,246	69,562
ataque =	porto_destino_perigoso = conhecido	94,484	69,527
ataque =	porto_destino_perigoso = conhecido and protocolo = tcp	94,484	69,527
altura = madrugada	servidor = mail and protocolo = tcp	64,246	68,892
RecordCount_perigoso = >=500	servidor = mail	65,050	68,702
altura = madrugada	servidor = mail	65,050	68,415

altura = madrugada	ataque = and porto_destino_perigoso = conhecido	65,692	66,535
altura = madrugada	ataque = and porto_destino_perigoso = conhecido and protocolo = tcp	65,692	66,535
servidor = mail	porto_destino_perigoso = conhecido	94,484	65,924
servidor = mail	porto_destino_perigoso = conhecido and protocolo = tcp	94,484	65,924
ataque =	servidor = mail	65,050	65,430
servidor = mail	protocolo = tcp	98,473	65,242
ataque =	servidor = mail and protocolo = tcp	64,246	65,212
altura = madrugada	ataque = and protocolo = tcp	69,681	64,416
altura = madrugada	ataque =	70,875	64,156
ataque =	servidor = mail and porto_destino_perigoso = conhecido	62,288	64,119
ataque =	servidor = mail and porto_destino_perigoso = conhecido and protocolo = tcp	62,288	64,119
altura = madrugada	porto_destino_perigoso = conhecido	94,484	61,849
altura = madrugada	porto_destino_perigoso = conhecido and protocolo = tcp	94,484	61,849
servidor = mail	ataque = and porto_destino_perigoso = conhecido	65,692	60,796
servidor = mail	ataque = and porto_destino_perigoso = conhecido and protocolo = tcp	65,692	60,796
SO_Origem_Disc = vazio	ataque =	70,875	60,775
SO_Origem_Disc = vazio	ataque = and porto_destino_perigoso = conhecido	65,692	60,599
SO_Origem_Disc = vazio	ataque = and porto_destino_perigoso = conhecido and protocolo = tcp	65,692	60,599
altura = madrugada	protocolo = tcp	98,473	60,540
SO_Origem_Disc = vazio	ataque = and protocolo = tcp	69,681	60,243
servidor = mail	ataque = and protocolo = tcp	69,681	60,126
servidor = mail	ataque =	70,875	60,053

Resultados da aplicação do algoritmo K-Means (para 3 clusters com os parâmetros protocolo, ataque, porta origem, porta destino e número de acessos no mesmo dia) e Apriori aos dados recolhidos no ICBAS

Consequent	Antecedent	Support %	Confidence %
porto_destino_perigoso = conhecido	servidor = web	64,273	90,868
porto_destino_perigoso = conhecido	SO_Origem_Disc = vazio	61,648	88,971
porto_destino_perigoso = conhecido	ataque =	84,141	86,114
ataque =	SO_Origem_Disc = vazio	61,648	84,743
ataque =	servidor = web	64,273	82,842
ataque =	porto_destino_perigoso = conhecido	88,316	82,043
servidor = web	porto_destino_perigoso = conhecido	88,316	66,130
servidor = web	ataque = and porto_destino_perigoso = conhecido	72,457	65,384
servidor = web	SO_Origem_Disc = vazio	61,648	63,779
servidor = web	ataque =	84,141	63,280
SO_Origem_Disc = vazio	ataque = and porto_destino_perigoso = conhecido	72,457	62,717
SO_Origem_Disc = vazio	porto_destino_perigoso = conhecido	88,316	62,105
SO_Origem_Disc = vazio	ataque =	84,141	62,088
SO_Origem_Disc = vazio	servidor = web	64,273	61,174

Resultados da aplicação do algoritmo K-Means (para 2 clusters com os parâmetros sistema operativo e serviço emulados) e Apriori aos dados recolhidos no ICBAS

Consequent	**Antecedent**	**Support %**	**Confidence %**
protocolo = tcp	porto_destino_perigoso = conhecido	93,926	100,000
protocolo = tcp	so_destino = linux and porto_destino_perigoso = conhecido	65,829	100,000
protocolo = tcp	ataque = and porto_destino_perigoso = conhecido	67,530	100,000
protocolo = tcp	servidor = mail and porto_destino_perigoso = conhecido	70,992	100,000
protocolo = tcp	so_destino = linux	69,765	98,392
protocolo = tcp	servidor = mail	75,087	98,336
protocolo = tcp	ataque =	73,243	98,089
porto_destino_perigoso = conhecido	servidor = mail and protocolo = tcp	73,837	96,147
porto_destino_perigoso = conhecido	so_destino = linux and protocolo = tcp	68,644	95,899
porto_destino_perigoso = conhecido	protocolo = tcp	98,239	95,610
porto_destino_perigoso = conhecido	servidor = mail	75,087	94,547
porto_destino_perigoso = conhecido	so_destino = linux	69,765	94,358
porto_destino_perigoso = conhecido	ataque = and protocolo = tcp	71,843	93,997
porto_destino_perigoso = conhecido	ataque =	73,243	92,200
ataque =	so_destino = linux and protocolo = tcp	68,644	77,763

ataque =	so_destino = linux	69,765	77,743
ataque =	so_destino = linux and porto_destino_perigoso = conhecido	65,829	76,812
ataque =	so_destino = linux and porto_destino_perigoso = conhecido and protocolo = tcp	65,829	76,812
servidor = mail	porto_destino_perigoso = conhecido	93,926	75,583
servidor = mail	porto_destino_perigoso = conhecido and protocolo = tcp	93,926	75,583
servidor = mail	protocolo = tcp	98,239	75,161
so_destino = linux	ataque = and porto_destino_perigoso = conhecido	67,530	74,877
so_destino = linux	ataque = and porto_destino_perigoso = conhecido and protocolo = tcp	67,530	74,877
so_destino = linux	ataque = and protocolo = tcp	71,843	74,301
so_destino = linux	ataque =	73,243	74,052
ataque =	protocolo = tcp	98,239	73,131
ataque =	porto_destino_perigoso = conhecido	93,926	71,897
ataque =	porto_destino_perigoso = conhecido and protocolo = tcp	93,926	71,897
servidor = mail	ataque = and porto_destino_perigoso = conhecido	67,530	71,534
servidor = mail	ataque = and porto_destino_perigoso = conhecido and protocolo = tcp	67,530	71,534
servidor = mail	ataque =	73,243	71,236
servidor = mail	ataque = and protocolo = tcp	71,843	71,200
so_destino = linux	porto_destino_perigoso = conhecido	93,926	70,086
so_destino = linux	porto_destino_perigoso = conhecido and protocolo = tcp	93,926	70,086
so_destino = linux	protocolo = tcp	98,239	69,874
ataque =	servidor = mail	75,087	69,487
ataque =	servidor = mail and protocolo = tcp	73,837	69,276
ataque =	servidor = mail and porto_destino_perigoso = conhecido	70,992	68,045
ataque =	servidor = mail and porto_destino_perigoso = conhecido and protocolo = tcp	70,992	68,045
servidor = mail	so_destino = linux and porto_destino_perigoso = conhecido	65,829	65,161
servidor = mail	so_destino = linux and porto_destino_perigoso = conhecido and protocolo = tcp	65,829	65,161
servidor = mail	so_destino = linux and protocolo = tcp	68,644	64,452
servidor = mail	so_destino = linux	69,765	64,290
so_destino = linux	servidor = mail and porto_destino_perigoso = conhecido	70,992	60,422
so_destino = linux	servidor = mail and porto_destino_perigoso = conhecido and protocolo = tcp	70,992	60,422
RecordCount_perigoso = >=500	servidor = mail and porto_destino_perigoso = conhecido	70,992	60,115
RecordCount_perigoso = >=500	servidor = mail and porto_destino_perigoso = conhecido and protocolo = tcp	70,992	60,115

Apêndice B

Exemplo de configuração do Snort

```
#--------------------------------------------------
#   http://www.snort.org     Snort 2.8.5.1 Ruleset
#     Contact: snort-sigs@lists.sourceforge.net
#
#--------------------------------------------------
# $Id$
#
##################################################
# This file contains a sample snort configuration.
# You can take the following steps to create your own custom configuration:
#
#  1) Set the variables for your network
#  2) Configure dynamic loaded libraries
#  3) Configure preprocessors
#  4) Configure output plugins
#  5) Add any runtime config directives
#  6) Customize your rule set
#
##################################################
# Step #1: Set the network variables:
#
# You must change the following variables to reflect your local network. The
# variable is currently setup for an RFC 1918 address space.
#
# You can specify it explicitly as:
#
# var HOME_NET 10.1.1.0/24
#
# if Snort is built with IPv6 support enabled (--enable-ipv6), use:
#
# ipvar HOME_NET 10.1.1.0/24
#
# or use global variable $<interfacename>_ADDRESS which will be always
# initialized to IP address and netmask of the network interface which you run
# snort at.  Under Windows, this must be specified as
```

```
# $(<interfacename>_ADDRESS), such as:
# $(\Device\Packet_{12345678-90AB-CDEF-1234567890AB}_ADDRESS)
#
# var HOME_NET $eth0_ADDRESS
#
# You can specify lists of IP addresses for HOME_NET
# by separating the IPs with commas like this:
#
# var HOME_NET [10.1.1.0/24,192.168.1.0/24]
#
# MAKE SURE YOU DON'T PLACE ANY SPACES IN YOUR LIST!
#
# or you can specify the variable to be any IP address
# like this:
#-- alteração -- especifica a rede ou conjunto de IPs que queremos monitorizar
var HOME_NET [192.168.1.82/32,192.168.1.83/32,192.168.1.84/32,192.168.1.85/32]

# Set up the external network addresses as well.  A good start may be "any"
var EXTERNAL_NET any

# Configure your server lists.  This allows snort to only look for attacks to
# systems that have a service up.  Why look for HTTP attacks if you are not
# running a web server?  This allows quick filtering based on IP addresses
# These configurations MUST follow the same configuration scheme as defined
# above for $HOME_NET.

# List of DNS servers on your network
var DNS_SERVERS $HOME_NET

# List of SMTP servers on your network
var SMTP_SERVERS $HOME_NET

# List of web servers on your network
var HTTP_SERVERS $HOME_NET

# List of sql servers on your network
var SQL_SERVERS $HOME_NET

# List of telnet servers on your network
var TELNET_SERVERS $HOME_NET

# List of telnet servers on your network
var FTP_SERVERS $HOME_NET

# List of snmp servers on your network
var SNMP_SERVERS $HOME_NET

# Configure your service ports.  This allows snort to look for attacks destined
# to a specific application only on the ports that application runs on.  For
# example, if you run a web server on port 8180, set your HTTP_PORTS variable
# like this:
#
# portvar HTTP_PORTS 8180
```

```
#
# Ports you run web servers on
portvar HTTP_PORTS 80

# NOTE:  If you wish to define multiple HTTP ports, use the portvar
# syntax to represent lists of ports and port ranges.  Examples:
## portvar HTTP_PORTS [80,8080]
## portvar HTTP_PORTS [80,8000:8080]
# And only include the rule that uses $HTTP_PORTS once.
#
# The pre-2.8.0 approach of redefining the variable to a different port and
# including the rules file twice is obsolete.  See README.variables for more
# details.

# Ports you want to look for SHELLCODE on.
portvar SHELLCODE_PORTS !80

# Ports you might see oracle attacks on
portvar ORACLE_PORTS 1521

# Ports for FTP servers
portvar FTP_PORTS 21

# other variables
#
# AIM servers.  AOL has a habit of adDINg new AIM servers, so instead of
# modifying the signatures when they do, we add them to this list of servers.
var AIM_SERVERS [64.12.24.0/23,64.12.28.0/23,64.12.161.0/24,64.12.163.0/24,64.12.200.0/24,
  205.188.3.0/24,205.188.5.0/24,205.188.7.0/24,205.188.9.0/24,205.188.153.0/24,
    205.188.179.0/24,205.188.248.0/24]

# Path to your rules files (this can be a relative path)
# Note for Windows users:  You are advised to make this an absolute path,
# such as:  c:\snort\rules

# -- alteração -- especifica o directório onde se encontram as regras
var RULE_PATH /etc/snort/rules
var PREPROC_RULE_PATH ../preproc_rules

# Configure the snort decoder
# ============================
#
# Snort's decoder will alert on lots of things such as header
# truncation or options of unusual length or infrequently used tcp options
#
#
# Stop generic decode events:
#
# config disable_decode_alerts
#
# Stop Alerts on experimental TCP options
#
# config disable_tcpopt_experimental_alerts
```

```
#
# Stop Alerts on obsolete TCP options
#
# config disable_tcpopt_obsolete_alerts
#
# Stop Alerts on T/TCP alerts
#
# In snort 2.0.1 and above, this only alerts when a TCP option is detected
# that shows T/TCP being actively used on the network.  If this is normal
# behavior for your network, disable the next option.
#
# config disable_tcpopt_ttcp_alerts
#
# Stop Alerts on all other TCPOption type events:
#
# config disable_tcpopt_alerts
#
# Stop Alerts on invalid ip options
#
# config disable_ipopt_alerts
#
# Alert if value in length field (IP, TCP, UDP) is greater than the
# actual length of the captured portion of the packet that the length
# is supposed to represent:
#
# config enable_decode_oversized_alerts
#
# Same as above, but drop packet if in Inline mode -
# enable_decode_oversized_alerts must be enabled for this to work:
#
# config enable_decode_oversized_drops
#

# Configure the detection engine
# ==============================
#
# Use a different pattern matcher in case you have a machine with very limited
# resources:
#
# config detection: search-method lowmem

# -- alteração -- especifica o algoritmo a usar para detecção de tentativas de intrusão
config detection: search-method ac-bnfa # - O valor usado na configuração é o 3° mais
                        # eficiente porque os dois primeiros sobrecarregavam o processador.
config show_year

# Configure Inline Resets
# =======================
#
# If running an iptables firewall with snort in InlineMode() we can now
# perform resets via a physical device. We grab the indev from iptables
# and use this for the interface on which to send resets. This config
# option takes an argument for the src mac address you want to use in the
```

```
# reset packet.  This way the bridge can remain stealthy. If the src mac
# option is not set we use the mac address of the indev device. If we
# don't set this option we will default to sending resets via raw socket,
# which needs an ipaddress to be assigned to the int.
#
# config layer2resets: 00:06:76:DD:5F:E3

###################################################
# Step #2: Configure dynamic loaded libraries
#
# If snort was configured to use dynamically loaded libraries,
# those libraries can be loaded here.
#
# Each of the following configuration options can be done via
# the command line as well.
#
# Load all dynamic preprocessors from the install path
# (same as command line option --dynamic-preprocessor-lib-dir)
#
dynamicpreprocessor directory /usr/lib/snort/dynamicpreprocessor/
#
# Load a specific dynamic preprocessor library from the install path
# (same as command line option --dynamic-preprocessor-lib)
#
# dynamicpreprocessor file /usr/local/lib/snort_dynamicpreprocessor/libdynamicexample.so
#
# Load a dynamic engine from the install path
# (same as command line option --dynamic-engine-lib)
#
dynamicengine /usr/lib/snort/dynamicengine/libsf_engine.so
#
# Load all dynamic rules libraries from the install path
# (same as command line option --dynamic-detection-lib-dir)
#
# dynamicdetection directory /usr/local/lib/snort_dynamicrule/
#
# Load a specific dynamic rule library from the install path
# (same as command line option --dynamic-detection-lib)
#
# dynamicdetection file /usr/local/lib/snort_dynamicrule/libdynamicexamplerule.so
#

###################################################
# Step #3: Configure preprocessors
#
# General configuration for preprocessors is of
# the form
# preprocessor <name_of_processor>: <configuration_options>

# frag3: Target-based IP defragmentation
# --------------------------------------
#
# Frag3 is a brand new IP defragmentation preprocessor that is capable of
```

```
# performing "target-based" processing of IP fragments.  Check out the
# README.frag3 file in the doc directory for more background and configuration
# information.
#
# Frag3 configuration is a two step process, a global initialization phase
# followed by the definition of a set of defragmentation engines.
#
# Global configuration defines the number of fragmented packets that Snort can
# track at the same time and gives you options regarding the memory cap for the
# subsystem or, optionally, allows you to preallocate all the memory for the
# entire frag3 system.
#
# frag3_global options:
#   max_frags: Maximum number of frag trackers that may be active at once.
#              Default value is 8192.
#   memcap: Maximum amount of memory that frag3 may access at any given time.
#           Default value is 4MB.
#   prealloc_frags: Maximum number of individual fragments that may be processed
#                   at once.  This is instead of the memcap system, uses static
#                   allocation to increase performance.  No default value.  Each
#                   preallocated fragment typically eats ~1550 bytes.  However,
#                   the exact amount is determined by the snaplen, and this can
#                   go as high as 64K so beware!
#
# Target-based behavior is attached to an engine as a "policy" for handling
# overlaps and retransmissions as enumerated in the Paxson paper.  There are
# currently five policy types available: "BSD", "BSD-right", "First", "Linux"
# and "Last".  Engines can be bound to standard Snort CIDR blocks or
# IP lists.
#
# frag3_engine options:
#   timeout: Amount of time a fragmented packet may be active before expiring.
#            Default value is 60 seconds.
#   ttl_limit: Limit of delta allowable for TTLs of packets in the fragments.
#              Based on the initial received fragment TTL.
#   min_ttl: Minimum acceptable TTL for a fragment, frags with TTLs below this
#            value will be discarded.  Default value is 0.
#   detect_anomalies: Activates frag3's anomaly detection mechanisms.
#   policy: Target-based policy to assign to this engine.  Default is BSD.
#   bind_to: IP address set to bind this engine to.  Default is all hosts.
#
# Frag3 configuration example:
#preprocessor frag3_global: max_frags 65536, prealloc_frags 65536
#preprocessor frag3_engine: policy linux \
#                           bind_to [10.1.1.12/32,10.1.1.13/32] \
#                           detect_anomalies
#preprocessor frag3_engine: policy first \
#                           bind_to 10.2.1.0/24 \
#                           detect_anomalies
#preprocessor frag3_engine: policy last \
#                           bind_to 10.3.1.0/24
#preprocessor frag3_engine: policy bsd
```

```
preprocessor frag3_global: max_frags 65536
preprocessor frag3_engine: policy first detect_anomalies overlap_limit 10

# stream5: Target Based stateful inspection/stream reassembly for Snort
# ---------------------------------------------------------------------
# Stream5 is a target-based stream engine for Snort.  It handles both
# TCP and UDP connection tracking as well as TCP reassembly.
#
# See README.stream5 for details on the configuration options.
#
# Example config
preprocessor stream5_global: max_tcp 8192, track_tcp yes, \
                             track_udp no
preprocessor stream5_tcp: policy first, use_static_footprint_sizes, detect_anomalies,
  check_session_hijacking
# preprocessor stream5_udp: ignore_any_rules


# Performance Statistics
# ----------------------
# Documentation for this is provided in the Snort Manual.  You should read it.
# It is included in the release distribution as doc/snort_manual.pdf
#
# preprocessor perfmonitor: time 300 file /var/snort/snort.stats pktcnt 10000

# http_inspect: normalize and detect HTTP traffic and protocol anomalies
#
# lots of options available here. See doc/README.http_inspect.
# unicode.map should be wherever your snort.conf lives, or given
# a full path to where snort can find it.
preprocessor http_inspect: global \
    iis_unicode_map unicode.map 1252

preprocessor http_inspect_server: server default \
# -- alteração --
# profile all ports { 80 8080 8180 } # - especifica quais as portas a analisar para
# um determinado tipo de tráfego. Neste caso esta opção foi comentada para que todas
# as portas pudessem ser analisadas independentemente do tipo de tráfego.
oversize_dir_length 300 \
server_flow_depth 0 \ # especifica número de bytes a inspeccionar. Neste caso todo o
                      # tráfego é inspeccionado
ascii yes \ # - descodificar ou não caracteres ascii encriptados
utf_8 yes \ # - detectar ou não sequências UTF-8 que se apresentem no URI
u_encode yes \ # - emula o esquema de codificação do IIS
bare_byte yes \ # - permite interpretar codificações menos vulgares
base36 no \ # - permite descodificar caracteres codificados com base36. Está
            #   desactivado porque conflitua com a opção utf_8.
iis_unicode yes \ # - permite descodificar caracteres não pertencentes ao código ASCII
                  #   que sejam usados pelo IISm assim como pedidos UTF-8
double_decode yes \ # - emula funcionalidades do IIS. Deve estar activo quando ascii
                    #   está activo.
multi_slash yes \ # - identifica uso de mais do que uma barra seguida, por exemplo,
                  #   "‘foo////bar"’
```

```
iis_backslash yes \ # - normaliza barras para trás para barras para a frente, exemplo:
                    #   "‘/foo\bar"’ -> "‘/foo/bar"’
directory yes \ # - normaliza directórios transversais e auto-referenciados, exemplo:
                #   "‘/foo/fake\_dir/../bar"’ -> "‘/foo/bar"’
apache_whitespace yes \ # - ideal para quando se usa/emula apache
webroot yes \ # - alerta quando o pedido passa a baixo da raiz de directórios do
              #   servidor web
normalize_cookies # - normaliza cookies HTTP que estejam encriptados

#
#  Example unique server configuration
#
#preprocessor http_inspect_server: server 1.1.1.1 \
#    ports { 80 3128 8080 } \
#    server_flow_depth 0 \
#    ascii no \
#    double_decode yes \
#    non_rfc_char { 0x00 } \
#    chunk_length 500000 \
#    non_strict \
#    oversize_dir_length 300 \
#    no_alerts


# rpc_decode: normalize RPC traffic
# ---------------------------------
# RPC may be sent in alternate encodings besides the usual 4-byte encoding
# that is used by default. This plugin takes the port numbers that RPC
# services are running on as arguments - it is assumed that the given ports
# are actually running this type of service. If not, change the ports or turn
# it off.
# The RPC decode preprocessor uses generator ID 106
#
# arguments: space separated list
# alert_fragments - alert on any rpc fragmented TCP data
# no_alert_multiple_requests - don’t alert when >1 rpc query is in a packet
# no_alert_large_fragments - don’t alert when the fragmented
#                            sizes exceed the current packet size
# no_alert_incomplete - don’t alert when a single segment
#                       exceeds the current packet size

preprocessor rpc_decode: 111 32771

# bo: Back Orifice detector
# -------------------------
# Detects Back Orifice traffic on the network.
#
# arguments:
#   syntax:
#     preprocessor bo: noalert { client | server | general | snort_attack } \
#                      drop    { client | server | general | snort_attack }
#   example:
#     preprocessor bo: noalert { general server } drop { snort_attack }
```

```
#
#
# The Back Orifice detector uses Generator ID 105 and uses the
# following SIDS for that GID:
#  SID     Event description
# -----   ------------------
#   1       Back Orifice traffic detected
#   2       Back Orifice Client Traffic Detected
#   3       Back Orifice Server Traffic Detected
#   4       Back Orifice Snort Buffer Attack

preprocessor bo

# ftp_telnet: FTP & Telnet normalizer, protocol enforcement and buff overflow
# ------------------------------------------------------------------------
# This preprocessor normalizes telnet negotiation strings from telnet and
# ftp traffic.  It looks for traffic that breaks the normal data stream
# of the protocol, replacing it with a normalized representation of that
# traffic so that the "content" pattern matching keyword can work without
# requiring modifications.
#
# It also performs protocol correctness checks for the FTP command channel,
# and identifies open FTP data transfers.
#
# FTPTelnet has numerous options available, please read
# README.ftptelnet for help configuring the options for the global
# telnet, ftp server, and ftp client sections for the protocol.

#####
# Per Step #2, set the following to load the ftptelnet preprocessor
# dynamicpreprocessor file <full path to libsf_ftptelnet_preproc.so>
# or use commandline option
# --dynamic-preprocessor-lib <full path to libsf_ftptelnet_preproc.so>

preprocessor ftp_telnet: global \
   encrypted_traffic yes \ # - permite a detecção em canais encriptados
# -- alteração --
check_encrypted \ # - intrui o preprocessador para continuar a verificação durante uma
                  #   sessão encriptada
   inspection_type stateful # - operar ou não em modo orientado à conexão

preprocessor ftp_telnet_protocol: telnet \
   normalize \ # - normaliza o tráfego telnet
   ayt_attack_thresh 10 \ # - define o número de pedidos consecutivos de "‘Are You There"’
                          #   (AYT) necessários para gerar alertas.
# -- alteração --
   detect_anomalies # - detecta quando um subnegotiation begin (SB) não tem um correspondente
                    #   subnegotioation end (SE).

# This is consistent with the FTP rules as of 18 Sept 2004.
# CWD can have param length of 200
# MODE has an additional mode of Z (compressed)
# Check for string formats in USER & PASS commands
```

```
# Check nDTM commands that set modification time on the file.
preprocessor ftp_telnet_protocol: ftp server default \
# -- alteração --
   alt_max_param_len 200 { CWD } \ # - define o tamanho máximo em bytes para os parâmetros
                                   #   dos comandos FTP especificados
   cmd_validity MODE < char ASBCZ > \ # - definição do formato válido para um determinado
                                      #   comando
   cmd_validity MDTM < [ date nnnnnnnnnnnnnn[.n[n[n]]] ] string > \
   chk_str_fmt { USER PASS RNFR RNTO SITE MKD } \ # - define inspecção de ataques baseados
                                                  #   no formato do texto
   telnet_cmds yes \ # - activa detecção de sequências de escape de telnet no canal de
                     #   comandos FTP

preprocessor ftp_telnet_protocol: ftp client default \
   max_resp_len 256 \
   bounce yes \
   telnet_cmds yes

# smtp: SMTP normalizer, protocol enforcement and buffer overflow
# ---------------------------------------------------------------------------
# This preprocessor normalizes SMTP commands by removing extraneous spaces.
# It looks for overly long command lines, response lines, and data header lines.
# It can alert on invalid commands, or specific valid commands.  It can optionally
# ignore mail data, and can ignore TLS encrypted data.
#
# SMTP has numerous options available, please read README.SMTP for help
# configuring options.

#####
# Per Step #2, set the following to load the smtp preprocessor
# dynamicpreprocessor file <full path to libsf_smtp_preproc.so>
# or use commandline option
# --dynamic-preprocessor-lib <full path to libsf_smtp_preproc.so>

preprocessor smtp: \
# -- alteração --
#  ports { 25 587 691 } \ # - especifica portas para a detecção em tráfego SMTP. Está
                          # comentado para que este tráfego seja inspeccionado em qualquer
                          # porta.
  inspection_type stateful \ # - define se é ou não orientado à conexão
  normalize all \ # - activa a normalização para todos os comandos
#  normalize_cmds { EXPN VRFY RCPT } \
  alt_max_command_line_len 260 { MAIL } \# - espefica um tamanho máximo para um determinado
                                         #   comando SMTP.
  alt_max_command_line_len 300 { RCPT } \
  alt_max_command_line_len 500 { HELP HELO ETRN } \
  alt_max_command_line_len 255 { EXPN VRFY } \
  ignore_data \ # - não inspeccionar o conteúdo dos emails
  max_command_line_len 512 \ # - definir tamanho máximo para um comando SMTP (valor
                             #   recomendado pelo RFC 2821).
  max_header_line_len 1024 \ # - definir tamanho máximo para os cabeçalhos de tráfego SMTP
                             #   (valor recomendado pelo RFC 2821).
  max_response_line_len 512 \ # - definir tamanho máximo para uma resposta SMTP (valor
```

```
                              #   recomendado pelo RFC 2821).
  alert_unknown_cmds \ # - alerta caso não conheça o comando
  xlink2state { enable } # - activar alertas xlink2state. Activado por omissão.

# sfPortscan
# ----------
# Portscan detection module.  Detects various types of portscans and
# portsweeps.  For more information on detection philosophy, alert types,
# and detailed portscan information, please refer to the README.sfportscan.
#
# -configuration options-
#     proto { tcp udp icmp ip all }
#       The arguments to the proto option are the types of protocol scans that
#       the user wants to detect.  Arguments should be separated by spaces and
#       not commas.
#     scan_type { portscan portsweep decoy_portscan distributed_portscan all }
#       The arguments to the scan_type option are the scan types that the
#       user wants to detect.  Arguments should be separated by spaces and not
#       commas.
#     sense_level { low|medium|high }
#       There is only one argument to this option and it is the level of
#       sensitivity in which to detect portscans.  The 'low' sensitivity
#       detects scans by the common method of looking for response errors, such
#       as TCP RSTs or ICMP unreachables.  This level requires the least
#       tuning.  The 'medium' sensitivity level detects portscans and
#       filtered portscans (portscans that receive no response).  This
#       sensitivity level usually requires tuning out scan events from NATed
#       IPs, DNS cache servers, etc.  The 'high' sensitivity level has
#       lower thresholds for portscan detection and a longer time window than
#       the 'medium' sensitivity level.  Requires more tuning and may be noisy
#       on very active networks.  However, this sensitivity levels catches the
#       most scans.
#     memcap { positive integer }
#       The maximum number of bytes to allocate for portscan detection.  The
#       higher this number the more nodes that can be tracked.
#     logfile { filename }
#       This option specifies the file to log portscan and detailed portscan
#       values to.  If there is not a leading /, then snort logs to the
#       configured log directory.  Refer to README.sfportscan for details on
#       the logged values in the logfile.
#     watch_ip { Snort IP List }
#     ignore_scanners { Snort IP List }
#     ignore_scanned { Snort IP List }
#       These options take a snort IP list as the argument.  The 'watch_ip'
#       option specifies the IP(s) to watch for portscan.  The
#       'ignore_scanners' option specifies the IP(s) to ignore as scanners.
#       Note that these hosts are still watched as scanned hosts.  The
#       'ignore_scanners' option is used to tune alerts from very active
#       hosts such as NAT, nessus hosts, etc.  The 'ignore_scanned' option
#       specifies the IP(s) to ignore as scanned hosts.  Note that these hosts
#       are still watched as scanner hosts.  The 'ignore_scanned' option is
#       used to tune alerts from very active hosts such as syslog servers, etc.
#     detect_ack_scans
```

```
#       This option will include sessions picked up in midstream by the stream
#       module, which is necessary to detect ACK scans.  However, this can lead to
#       false alerts, especially under heavy load with dropped packets; which is why
#       the option is off by default.
#
preprocessor sfportscan: proto  { all } \
                         memcap { 10000000 } \
# -- alteração --
 scan_type { all } \ # - analisa qualquer protocolo
       sense_level { medium } \ # - sensibilidade da detecção. Inicialmente usou-se o valor
                                #   High que procura continuamente dispositivos na rede e
                                #   gera estatísticas sobre o varrimento de portas efectuado
                                #   por eles, mas verificou-se que traz mais desvantagens do
                                #   que vantagens neste caso devido ao elevado número de
                                #   falsos positivos gerado.
 include_midstream \ # - verificar sessões recuperadas
 detect_ack_scans # - opção necessária à detecção de varrimentos de ACK

# arpspoof
#--------------------------------------
# Experimental ARP detection code from Jeff Nathan, detects ARP attacks,
# unicast ARP requests, and specific ARP mapping monitoring.  To make use of
# this preprocessor you must specify the IP and hardware address of hosts on
# the same layer 2 segment as you.  Specify one host IP MAC combo per line.
# Also takes a "-unicast" option to turn on unicast ARP request detection.
# Arpspoof uses Generator ID 112 and uses the following SIDS for that GID:

#  SID     Event description
# -----   ------------------
#   1       Unicast ARP request
#   2       Etherframe ARP mismatch (src)
#   3       Etherframe ARP mismatch (dst)
#   4       ARP cache overwrite attack

#preprocessor arpspoof
#preprocessor arpspoof_detect_host: 192.168.40.1 f0:0f:00:f0:0f:00

# ssh
# ----------------------------
# The SSH preprocessor detects the following exploits: Challenge-Response
# Authentication overflow, CRC 32 overflow, Secure CRT version string overflow,
# and protocol version mismatches.
#
# Both Challenge-Response Auth and CRC 32 attacks occur after the key exchange,
# and are therefore encrypted.  Both attacks involve sending a large payload
# (20kb+) to the server immediately after the authentication challenge.
# To detect the attacks, the SSH preprocessor counts the number of bytes
# transmitted to the server.  If those bytes exceed a pre-defined limit,
# set by the option "max_client_bytes", an alert is generated. Since
# the Challenge-Response Auth overflow only affects SSHv2, while CRC 32 only
# affects SSHv1, the SSH version string exchange is used to distinguish
# the attacks.
#
```

```
# The Secure CRT and protocol mismatch exploits are observable before
# the key exchange.
#
# SSH has numerous options available, please read README.ssh for help
# configuring options.

#####
# Per Step #2, set the following to load the ssh preprocessor
# dynamicpreprocessor file <full path to libsf_ssh_preproc.so>
# or use commandline option
# --dynamic-preprocessor-lib <full path to libsf_ssh_preproc.so>
#
preprocessor ssh: \
# server_ports { 22 } \
                  max_client_bytes 19600 \
                  max_encrypted_packets 20 \
                  enable_respoverflow enable_ssh1crc32 \
                  enable_srvoverflow enable_protomismatch \
# -- alteração --
  enable_badmsgdir enable_paysize enable_recognition # - alertar quando o tráfego segue
                                                     # na direcção errada, apresenta
                                                     # tamanho de payload inválido ou
                                                     # não é SSH e circula na porta 22.

# DCE/RPC
#--------------------------------------
#
# The dcerpc preprocessor detects and decodes SMB and DCE/RPC traffic.
# It is primarily interested in DCE/RPC data, and only decodes SMB
# to get at the DCE/RPC data carried by the SMB layer.
#
# Currently, the preprocessor only handles reassembly of fragmentation
# at both the SMB and DCE/RPC layer.  Snort rules can be evaded by
# using both types of fragmentation; with the preprocessor enabled
# the rules are given a buffer with a reassembled SMB or DCE/RPC
# packet to examine.
#
# At the SMB layer, only fragmentation using WriteAndX is currently
# reassembled.  Other methods will be handled in future versions of
# the preprocessor.
#
# Autodetection of SMB is done by looking for "\xFFSMB" at the start of
# the SMB data, as well as checking the NetBIOS header (which is always
# present for SMB) for the type "SMB Session".
#
# Autodetection of DCE/RPC is not as reliable.  Currently, two bytes are
# checked in the packet.  Assuming that the data is a DCE/RPC header,
# one byte is checked for DCE/RPC version (5) and another for the type
# "DCE/RPC Request".  If both match, the preprocessor proceeds with that
# assumption that it is looking at DCE/RPC data.  If subsequent checks
# are nonsensical, it ends processing.
#
# DCERPC has numerous options available, please read README.dcerpc for help
```

```
# configuring options.

#####
# Per Step #2, set the following to load the dcerpc preprocessor
# dynamicpreprocessor file <full path to libsf_dcerpc_preproc.so>
# or use commandline option
# --dynamic-preprocessor-lib <full path to libsf_dcerpc_preproc.so>
#
#preprocessor dcerpc: \
#    autodetect \
#    max_frag_size 3000 \
#    memcap 100000


# DCE/RPC 2
#--------------------------------------
# See doc/README.dcerpc2 for explanations of what the
# preprocessor does and how to configure it.
#
# -- alteração -- detecta e descodifica tráfego SMB. Foi comentado para que o SNORT
#                 arrancasse e não é necessário neste caso.
#preprocessor dcerpc2
#preprocessor dcerpc2_server: default


# DNS
#--------------------------------------
# The dns preprocessor (currently) decodes DNS Response traffic
# and detects a few vulnerabilities.
#
# DNS has a few options available, please read README.dns for
# help configuring options.

#####
# Per Step #2, set the following to load the dns preprocessor
# dynamicpreprocessor file <full path to libsf_dns_preproc.so>
# or use commandline option
# --dynamic-preprocessor-lib <full path to libsf_dns_preproc.so>

preprocessor dns: \
# -- alteração --    ports { 53 } \ # - comentado para que a detecção de tráfego DNS
                                    #   ocorra em qualquer porta
    enable_rdata_overflow \ # - verifica RDATA TXT Overflow
    enable_obsolete_types \ # - alerta perante registos de tipo obsoleto
    enable_experimental_types # - alerta perante registos de tipo experimental


# SSL
#--------------------------------------
# Encrypted traffic should be ignored by Snort for both performance reasons
# and to reduce false positives.  The SSL Dynamic Preprocessor (SSLPP)
# inspects SSL traffic and optionally determines if and when to stop
# inspection of it.
```

```
#
# Typically, SSL is used over port 443 as HTTPS.  By enabling the SSLPP to
# inspect port 443, only the SSL handshake of each connection will be
# inspected.  Once the traffic is determined to be encrypted, no further
# inspection of the data on the connection is made.
#
# If you don't necessarily trust all of the SSL capable servers on your
# network, you should remove the "trustservers" option from the configuration.
#
#   Important note: Stream5 should be explicitly told to reassemble
#                   traffic on the ports that you intend to inspect SSL
#                   encrypted traffic on.
#
#   To add reassembly on port 443 to Stream5, use 'port both 443' in the
#   Stream5 configuration.

preprocessor ssl: noinspect_encrypted, trustservers


##################################################################
# Step #4: Configure output plugins
#
# Uncomment and configure the output plugins you decide to use.  General
# configuration for output plugins is of the form:
#
# output <name_of_plugin>: <configuration_options>
#
# alert_syslog: log alerts to syslog
# ----------------------------------
# Use one or more syslog facilities as arguments.  Win32 can also optionally
# specify a particular hostname/port.  Under Win32, the default hostname is
# '127.0.0.1', and the default port is 514.
#
# [Unix flavours should use this format...]
# output alert_syslog: LOG_AUTH LOG_ALERT
#
# [Win32 can use any of these formats...]
# output alert_syslog: LOG_AUTH LOG_ALERT
# output alert_syslog: host=hostname, LOG_AUTH LOG_ALERT
# output alert_syslog: host=hostname:port, LOG_AUTH LOG_ALERT

# log_tcpdump: log packets in binary tcpdump format
# -------------------------------------------------
# The only argument is the output file name.
#
# output log_tcpdump: tcpdump.log

# database: log to a variety of databases
# ---------------------------------------
# See the README.database file for more information about configuring
# and using this plugin.
#
# output database: log, mysql, user=root password=test dbname=db host=localhost
```

```
# output database: alert, postgresql, user=snort dbname=snort
# output database: log, odbc, user=snort dbname=snort
# output database: log, mssql, dbname=snort user=snort password=test
# output database: log, oracle, dbname=snort user=snort password=test

# unified: Snort unified binary format alerting and logging
# -------------------------------------------------------------
# The unified output plugin provides two new formats for logging and generating
# alerts from Snort, the "unified" format.  The unified format is a straight
# binary format for logging data out of Snort that is designed to be fast and
# efficient.  Used with barnyard (the new alert/log processor), most of the
# overhead for logging and alerting to various slow storage mechanisms such as
# databases or the network can now be avoided.
#
# Check out the spo_unified.h file for the data formats.
#
# Two arguments are supported.
#    filename - base filename to write to (current time_t is appended)
#    limit    - maximum size of spool file in MB (default: 128)
#
# output alert_unified: filename snort.alert, limit 128
# output log_unified: filename snort.log, limit 128


# prelude: log to the Prelude Hybrid IDS system
# -----------------------------------------
#
# profile = Name of the Prelude profile to use (default is snort).
#
# Snort priority to IDMEF severity mappings:
# high < medium < low < info
#
# These are the default mapped from classification.config:
# info   = 4
# low    = 3
# medium = 2
# high   = anything below medium
#
# output alert_prelude
# output alert_prelude: profile=snort-profile-name


# You can optionally define new rule types and associate one or more output
# plugins specifically to that type.
#
# This example will create a type that will log to just tcpdump.
# ruletype suspicious
# {
#   type log
#   output log_tcpdump: suspicious.log
# }
#
# EXAMPLE RULE FOR SUSPICIOUS RULETYPE:
```

```
# suspicious tcp $HOME_NET any -> $HOME_NET 6667 (msg:"Internal IRC Server";)
#
# This example will create a rule type that will log to syslog and a mysql
# database:
# ruletype redalert
# {
#   type alert
#   output alert_syslog: LOG_AUTH LOG_ALERT
#   output database: log, mysql, user=snort dbname=snort host=localhost
# }
#
# EXAMPLE RULE FOR REDALERT RULETYPE:
# redalert tcp $HOME_NET any -> $EXTERNAL_NET 31337 \
#   (msg:"Someone is being LEET"; flags:A+;)

#
# Include classification & priority settings
# Note for Windows users:  You are advised to make this an absolute path,
# such as:  c:\snort\etc\classification.config
#

include classification.config

#
# Include reference systems
# Note for Windows users:  You are advised to make this an absolute path,
# such as:  c:\snort\etc\reference.config
#

include reference.config

##################################################################
# Step #5: Configure snort with config statements
#
# See the snort manual for a full set of configuration references
#
# config flowbits_size: 64
#
# New global ignore_ports config option from Andy Mullican
#
# config ignore_ports: <tcp|udp> <list of ports separated by whitespace>
# config ignore_ports: tcp 21 6667:6671 1356
# config ignore_ports: udp 1:17 53


##################################################################
# Step #6: Customize your rule set
#
# Up to date snort rules are available at http://www.snort.org
#
# The snort web site has documentation about how to write your own custom snort
# rules.
```

```
#========================================
# Include all relevant rulesets here
#
# The following rulesets are disabled by default:
#
#   web-attacks, backdoor, shellcode, policy, porn, info, icmp-info, virus,
#   chat, multimedia, and p2p
#
# These rules are either site policy specific or require tuning in order to not
# generate false positive alerts in most enviornments.
#
# Please read the specific include file for more information and
# README.alert_order for how rule ordering affects how alerts are triggered.
#========================================

include $RULE_PATH/local.rules
include $RULE_PATH/bad-traffic.rules
include $RULE_PATH/exploit.rules
include $RULE_PATH/scan.rules
include $RULE_PATH/finger.rules
include $RULE_PATH/ftp.rules
include $RULE_PATH/telnet.rules
include $RULE_PATH/rpc.rules
include $RULE_PATH/rservices.rules
#include $RULE_PATH/dos.rules
include $RULE_PATH/ddos.rules
include $RULE_PATH/dns.rules
include $RULE_PATH/tftp.rules

include $RULE_PATH/web-cgi.rules
include $RULE_PATH/web-coldfusion.rules
include $RULE_PATH/web-iis.rules
include $RULE_PATH/web-frontpage.rules
include $RULE_PATH/web-misc.rules
include $RULE_PATH/web-client.rules
include $RULE_PATH/web-php.rules

include $RULE_PATH/sql.rules
include $RULE_PATH/x11.rules
include $RULE_PATH/icmp.rules
#include $RULE_PATH/netbios.rules
include $RULE_PATH/misc.rules
include $RULE_PATH/attack-responses.rules
include $RULE_PATH/oracle.rules
include $RULE_PATH/mysql.rules
include $RULE_PATH/snmp.rules

include $RULE_PATH/smtp.rules
include $RULE_PATH/imap.rules
include $RULE_PATH/pop2.rules
include $RULE_PATH/pop3.rules

include $RULE_PATH/nntp.rules
```

```
include $RULE_PATH/other-ids.rules
# -- alteração -- é necessário comentar algumas regras que já não são incluidas no pacote
                 #disponível no site oficial do SNORT
 include $RULE_PATH/web-attacks.rules
 include $RULE_PATH/backdoor.rules
 include $RULE_PATH/shellcode.rules
 include $RULE_PATH/policy.rules
 include $RULE_PATH/porn.rules
 include $RULE_PATH/info.rules
 include $RULE_PATH/icmp-info.rules
 include $RULE_PATH/virus.rules
 include $RULE_PATH/chat.rules
 include $RULE_PATH/multimedia.rules
 include $RULE_PATH/p2p.rules
# include $RULE_PATH/spyware-put.rules
# include $RULE_PATH/specific-threats.rules
include $RULE_PATH/experimental.rules

# include $PREPROC_RULE_PATH/preprocessor.rules
# include $PREPROC_RULE_PATH/decoder.rules

# Include any thresholding or suppression commands. See threshold.conf in the
# <snort src>/etc directory for details. Commands don't necessarily need to be
# contained in this conf, but a separate conf makes it easier to maintain them.
# Note for Windows users:  You are advised to make this an absolute path,
# such as:  c:\snort\etc\threshold.conf
# Uncomment if needed.
# include threshold.conf

----------------
```

Apêndice C

Excerto de ficheiro de *log* do Honeyd

```
2010-04-12-20:40:58.5714 tcp(6) - 41.225.140.245 1957 193.136.62.71 135: 48 S
2010-04-12-20:40:58.5759 tcp(6) - 41.225.140.245 1959 193.136.62.73 135: 48 S
2010-04-12-20:40:59.1902 tcp(6) - 41.225.140.245 1957 193.136.62.71 135: 48 S
2010-04-12-20:40:59.1905 tcp(6) - 41.225.140.245 1959 193.136.62.73 135: 48 S
2010-04-12-20:40:59.7881 tcp(6) - 41.225.140.245 1959 193.136.62.73 135: 48 S
2010-04-12-20:40:59.7891 tcp(6) - 41.225.140.245 1957 193.136.62.71 135: 48 S
2010-04-12-20:41:00.3653 tcp(6) - 41.225.140.245 1984 193.136.62.74 135: 48 S
2010-04-12-20:41:00.9925 tcp(6) - 41.225.140.245 1984 193.136.62.74 135: 48 S
2010-04-12-20:41:01.5896 tcp(6) - 41.225.140.245 1984 193.136.62.74 135: 48 S
2010-04-12-20:45:00.1826 udp(17) - 217.75.242.198 1133 193.136.62.74 1434: 404
2010-04-12-21:02:45.7561 tcp(6) - 94.236.10.10 80 193.136.62.74 3072: 40 SA
2010-04-12-21:12:52.7536 udp(17) - 119.153.72.60 44241 193.136.62.72 18293: 95
2010-04-12-21:22:08.0184 tcp(6) - 220.168.203.105 6000 193.136.62.71 2967: 40 S
2010-04-12-21:22:08.0185 tcp(6) - 220.168.203.105 6000 193.136.62.72 2967: 40 S
2010-04-12-21:22:08.0200 tcp(6) - 220.168.203.105 6000 193.136.62.73 2967: 40 S
2010-04-12-21:24:28.2151 udp(17) - 94.99.166.60 12338 193.136.62.72 18294: 95
2010-04-12-21:27:50.7026 tcp(6) - 94.236.10.10 80 193.136.62.74 3072: 40 SA
2010-04-12-21:37:04.7914 tcp(6) - 66.186.59.50 6667 193.136.62.71 1187: 44 SA
2010-04-12-21:46:02.0136 tcp(6) - 66.186.59.50 6667 193.136.62.74 1169: 44 SA
2010-04-12-21:49:46.7080 tcp(6) - 193.198.179.20 3458 193.136.62.71 135: 48 S [Windows XP SP1]
2010-04-12-21:49:48.9645 tcp(6) - 193.198.179.20 3458 193.136.62.71 135: 48 S [Windows XP SP1]
2010-04-12-22:01:44.4305 tcp(6) - 94.236.10.10 80 193.136.62.73 3072: 40 SA
2010-04-12-22:05:26.7695 tcp(6) - 77.92.142.70 80 193.136.62.72 3072: 44 SA
2010-04-12-22:11:07.0852 tcp(6) - 74.247.206.186 2738 193.136.62.74 445: 48 S [Windows XP SP1]
2010-04-12-22:11:07.8252 tcp(6) - 74.247.206.186 2738 193.136.62.74 445: 48 S [Windows XP SP1]
2010-04-12-22:11:08.4849 tcp(6) - 74.247.206.186 2738 193.136.62.74 445: 48 S [Windows XP SP1]
2010-04-12-22:13:05.9873 tcp(6) - 66.186.59.50 6667 193.136.62.74 1114: 44 SA
2010-04-12-22:17:53.8567 tcp(6) - 82.91.125.98 1135 193.136.62.71 3128: 48 S [Windows XP SP1]
2010-04-12-22:17:53.8634 tcp(6) - 82.91.125.98 1136 193.136.62.72 3128: 48 S [Windows XP SP1]
2010-04-12-22:17:53.8637 tcp(6) - 82.91.125.98 1137 193.136.62.73 3128: 48 S [Windows XP SP1]
2010-04-12-22:17:53.8640 tcp(6) - 82.91.125.98 1138 193.136.62.74 3128: 48 S [Windows XP SP1]
2010-04-12-22:17:54.5741 tcp(6) - 82.91.125.98 1135 193.136.62.71 3128: 48 S [Windows XP SP1]
2010-04-12-22:17:54.6826 tcp(6) - 82.91.125.98 1137 193.136.62.73 3128: 48 S [Windows XP SP1]
2010-04-12-22:17:54.6829 tcp(6) - 82.91.125.98 1138 193.136.62.74 3128: 48 S [Windows XP SP1]
2010-04-12-22:17:54.6873 tcp(6) - 82.91.125.98 1136 193.136.62.72 3128: 48 S [Windows XP SP1]
```

```
2010-04-12-22:17:55.4134 tcp(6) - 82.91.125.98 1138 193.136.62.74 3128: 48 S [Windows XP SP1]
2010-04-12-22:17:55.4204 tcp(6) - 82.91.125.98 1136 193.136.62.72 3128: 48 S [Windows XP SP1]
2010-04-12-22:17:55.5864 tcp(6) - 82.91.125.98 1135 193.136.62.71 3128: 48 S [Windows XP SP1]
2010-04-12-22:23:20.1819 tcp(6) - 193.198.179.20 3692 193.136.62.71 135: 48 S [Windows XP SP1]
2010-04-12-22:23:25.1529 tcp(6) - 193.198.179.20 3692 193.136.62.71 135: 48 S [Windows XP SP1]
2010-04-12-22:26:08.1196 tcp(6) - 41.225.140.245 1620 193.136.62.71 135: 48 S
2010-04-12-22:26:08.1205 tcp(6) - 41.225.140.245 1622 193.136.62.73 135: 48 S
2010-04-12-22:26:08.1208 tcp(6) - 41.225.140.245 1621 193.136.62.72 135: 48 S
2010-04-12-22:26:08.1269 tcp(6) - 41.225.140.245 1623 193.136.62.74 135: 48 S
2010-04-12-22:26:08.7453 tcp(6) - 41.225.140.245 1622 193.136.62.73 135: 48 S
2010-04-12-22:26:08.7456 tcp(6) - 41.225.140.245 1623 193.136.62.74 135: 48 S
2010-04-12-22:26:08.7503 tcp(6) - 41.225.140.245 1620 193.136.62.71 135: 48 S
2010-04-12-22:26:08.7506 tcp(6) - 41.225.140.245 1621 193.136.62.72 135: 48 S
2010-04-12-22:26:09.3471 tcp(6) - 41.225.140.245 1622 193.136.62.73 135: 48 S
2010-04-12-22:26:09.3485 tcp(6) - 41.225.140.245 1623 193.136.62.74 135: 48 S
2010-04-12-22:26:09.3510 tcp(6) - 41.225.140.245 1621 193.136.62.72 135: 48 S
```

Apêndice D

Excerto de ficheiro de *log* do Snort

```
[**] [1:486:5] ICMP Destination Unreachable Communication with Destination Host is
  Administratively Prohibited [**]
[Classification: Misc activity] [Priority: 3]
04/12/10-20:40:58.572447 0:C:29:8B:11:9B -> 0:19:56:3F:5D:E0 type:0x800 len:0x5A
193.136.62.71 -> 41.225.140.245 ICMP TTL:64 TOS:0xC0 ID:14412 IpLen:20 DgmLen:76
Type:3  Code:10  DESTINATION UNREACHABLE: ADMINISTRATIVELY PROHIBITED HOST FILTERED
** ORIGINAL DATAGRAM DUMP:
41.225.140.245:1957 -> 193.136.62.71:135 TCP TTL:42 TOS:0x0 ID:64754 IpLen:20 DgmLen:48 DF
Seq: 0x10E9409F
(20 more bytes of original packet)
** END OF DUMP

[**] [1:486:5] ICMP Destination Unreachable Communication with Destination Host is
  Administratively Prohibited [**]
[Classification: Misc activity] [Priority: 3]
04/12/10-20:40:58.572491 0:C:29:8B:11:9B -> 0:19:56:3F:5D:E0 type:0x800 len:0x5A
193.136.62.70 -> 41.225.140.245 ICMP TTL:64 TOS:0xC0 ID:14413 IpLen:20 DgmLen:76
Type:3  Code:10  DESTINATION UNREACHABLE: ADMINISTRATIVELY PROHIBITED HOST FILTERED
** ORIGINAL DATAGRAM DUMP:
41.225.140.245:1956 -> 193.136.62.70:135 TCP TTL:42 TOS:0x0 ID:64753 IpLen:20 DgmLen:48 DF
Seq: 0x10E87AC3
(20 more bytes of original packet)
** END OF DUMP

[**] [1:486:5] ICMP Destination Unreachable Communication with Destination Host is
  Administratively Prohibited [**]
[Classification: Misc activity] [Priority: 3]
04/12/10-20:40:58.572732 0:C:29:8B:11:9B -> 0:19:56:3F:5D:E0 type:0x800 len:0x5A
193.136.62.73 -> 41.225.140.245 ICMP TTL:64 TOS:0xC0 ID:14414 IpLen:20 DgmLen:76
Type:3  Code:10  DESTINATION UNREACHABLE: ADMINISTRATIVELY PROHIBITED HOST FILTERED
** ORIGINAL DATAGRAM DUMP:
41.225.140.245:1959 -> 193.136.62.73:135 TCP TTL:42 TOS:0x0 ID:64756 IpLen:20 DgmLen:48 DF
Seq: 0x10EACF3C
(20 more bytes of original packet)
** END OF DUMP
```

```
[**] [1:486:5] ICMP Destination Unreachable Communication with Destination Host is
  Administratively Prohibited [**]
[Classification: Misc activity] [Priority: 3]
04/12/10-20:40:59.189031 0:C:29:8B:11:9B -> 0:19:56:3F:5D:E0 type:0x800 len:0x5A
193.136.62.71 -> 41.225.140.245 ICMP TTL:64 TOS:0xC0 ID:14415 IpLen:20 DgmLen:76
Type:3  Code:10  DESTINATION UNREACHABLE: ADMINISTRATIVELY PROHIBITED HOST FILTERED
** ORIGINAL DATAGRAM DUMP:
41.225.140.245:1957 -> 193.136.62.71:135 TCP TTL:42 TOS:0x0 ID:64758 IpLen:20 DgmLen:48 DF
Seq: 0x10E9409F
(20 more bytes of original packet)
** END OF DUMP

[**] [1:486:5] ICMP Destination Unreachable Communication with Destination Host is
  Administratively Prohibited [**]
[Classification: Misc activity] [Priority: 3]
04/12/10-20:40:59.189125 0:C:29:8B:11:9B -> 0:19:56:3F:5D:E0 type:0x800 len:0x5A
193.136.62.73 -> 41.225.140.245 ICMP TTL:64 TOS:0xC0 ID:14416 IpLen:20 DgmLen:76
Type:3  Code:10  DESTINATION UNREACHABLE: ADMINISTRATIVELY PROHIBITED HOST FILTERED
** ORIGINAL DATAGRAM DUMP:
41.225.140.245:1959 -> 193.136.62.73:135 TCP TTL:42 TOS:0x0 ID:64757 IpLen:20 DgmLen:48 DF
Seq: 0x10EACF3C
(20 more bytes of original packet)
** END OF DUMP

[**] [1:486:5] ICMP Destination Unreachable Communication with Destination Host is
  Administratively Prohibited [**]
[Classification: Misc activity] [Priority: 3]
04/12/10-20:40:59.787486 0:C:29:8B:11:9B -> 0:19:56:3F:5D:E0 type:0x800 len:0x5A
193.136.62.73 -> 41.225.140.245 ICMP TTL:64 TOS:0xC0 ID:14417 IpLen:20 DgmLen:76
Type:3  Code:10  DESTINATION UNREACHABLE: ADMINISTRATIVELY PROHIBITED HOST FILTERED
** ORIGINAL DATAGRAM DUMP:
41.225.140.245:1959 -> 193.136.62.73:135 TCP TTL:42 TOS:0x0 ID:64779 IpLen:20 DgmLen:48 DF
Seq: 0x10EACF3C
(20 more bytes of original packet)
** END OF DUMP

[**] [1:486:5] ICMP Destination Unreachable Communication with Destination Host is
  Administratively Prohibited [**]
[Classification: Misc activity] [Priority: 3]
04/12/10-20:40:59.788661 0:C:29:8B:11:9B -> 0:19:56:3F:5D:E0 type:0x800 len:0x5A
193.136.62.71 -> 41.225.140.245 ICMP TTL:64 TOS:0xC0 ID:14418 IpLen:20 DgmLen:76
Type:3  Code:10  DESTINATION UNREACHABLE: ADMINISTRATIVELY PROHIBITED HOST FILTERED
** ORIGINAL DATAGRAM DUMP:
41.225.140.245:1957 -> 193.136.62.71:135 TCP TTL:42 TOS:0x0 ID:64780 IpLen:20 DgmLen:48 DF
Seq: 0x10E9409F
(20 more bytes of original packet)
** END OF DUMP

[**] [1:486:5] ICMP Destination Unreachable Communication with Destination Host is
  Administratively Prohibited [**]
[Classification: Misc activity] [Priority: 3]
04/12/10-22:17:55.416686 0:C:29:8B:11:9B -> 0:19:56:3F:5D:E0 type:0x800 len:0x5A
```

```
193.136.62.72 -> 82.91.125.98 ICMP TTL:64 TOS:0xC0 ID:65381 IpLen:20 DgmLen:76
Type:3  Code:10  DESTINATION UNREACHABLE: ADMINISTRATIVELY PROHIBITED HOST FILTERED
** ORIGINAL DATAGRAM DUMP:
82.91.125.98:1136 -> 193.136.62.72:3128 TCP TTL:104 TOS:0x0 ID:22596 IpLen:20 DgmLen:48 DF
Seq: 0xF5BEFE9E
(20 more bytes of original packet)
** END OF DUMP

[**] [1:486:5] ICMP Destination Unreachable Communication with Destination Host is
  Administratively Prohibited [**]
[Classification: Misc activity] [Priority: 3]
04/12/10-22:17:55.585206 0:C:29:8B:11:9B -> 0:19:56:3F:5D:E0 type:0x800 len:0x5A
193.136.62.71 -> 82.91.125.98 ICMP TTL:64 TOS:0xC0 ID:65382 IpLen:20 DgmLen:76
Type:3  Code:10  DESTINATION UNREACHABLE: ADMINISTRATIVELY PROHIBITED HOST FILTERED
** ORIGINAL DATAGRAM DUMP:
82.91.125.98:1135 -> 193.136.62.71:3128 TCP TTL:104 TOS:0x0 ID:22732 IpLen:20 DgmLen:48 DF
Seq: 0xB705A281
(20 more bytes of original packet)
** END OF DUMP

[**] [1:486:5] ICMP Destination Unreachable Communication with Destination Host is
  Administratively Prohibited [**]
[Classification: Misc activity] [Priority: 3]
04/12/10-22:18:03.160077 0:C:29:8B:11:9B -> 0:19:56:3F:5D:E0 type:0x800 len:0x5A
193.136.62.70 -> 82.91.125.98 ICMP TTL:64 TOS:0xC0 ID:65383 IpLen:20 DgmLen:76
Type:3  Code:10  DESTINATION UNREACHABLE: ADMINISTRATIVELY PROHIBITED HOST FILTERED
** ORIGINAL DATAGRAM DUMP:
82.91.125.98:1134 -> 193.136.62.70:3128 TCP TTL:104 TOS:0x0 ID:25919 IpLen:20 DgmLen:48 DF
Seq: 0xA5E6EAE2
(20 more bytes of original packet)
** END OF DUMP

[**] [1:486:5] ICMP Destination Unreachable Communication with Destination Host is
  Administratively Prohibited [**]
[Classification: Misc activity] [Priority: 3]
04/12/10-22:23:20.177453 0:C:29:8B:11:9B -> 0:19:56:3F:5D:E0 type:0x800 len:0x5A
193.136.62.71 -> 193.198.179.20 ICMP TTL:64 TOS:0xC0 ID:12721 IpLen:20 DgmLen:76
Type:3  Code:10  DESTINATION UNREACHABLE: ADMINISTRATIVELY PROHIBITED HOST FILTERED
** ORIGINAL DATAGRAM DUMP:
193.198.179.20:3692 -> 193.136.62.71:135 TCP TTL:111 TOS:0x0 ID:62851 IpLen:20 DgmLen:48 DF
Seq: 0xA46B08BC
(20 more bytes of original packet)
** END OF DUMP
```

Apêndice E

Excerto do ficheiro resultante da aplicação desenvolvida

```
data;hora;protocolo;inicio_fim;so_destino;servidor;ip_origem;porto_origem;ip_destino;
  porto_destino;so_origem;pais;insituicao;exposicao;ataque;diferenca
2010-04-12;20:40:58;tcp;-;windows;mail;41.225.140.245;1957;193.136.62.71;135;;TUNISIA;
  ISEP;publico;ICMP Destination Unreachable Communication with Destination Host is
    Administratively Prohibited;
2010-04-12;20:40:58;tcp;-;linux;mail;41.225.140.245;1959;193.136.62.73;135;;TUNISIA;
  ISEP;publico;ICMP Destination Unreachable Communication with Destination Host is
    Administratively Prohibited;0
2010-04-12;20:40:59;tcp;-;windows;mail;41.225.140.245;1957;193.136.62.71;135;;TUNISIA;
  ISEP;publico;ICMP Destination Unreachable Communication with Destination Host is
    Administratively Prohibited;0
2010-04-12;20:40:59;tcp;-;linux;mail;41.225.140.245;1959;193.136.62.73;135;;TUNISIA;
  ISEP;publico;ICMP Destination Unreachable Communication with Destination Host is
    Administratively Prohibited;0
2010-04-12;20:40:59;tcp;-;linux;mail;41.225.140.245;1959;193.136.62.73;135;;TUNISIA;
  ISEP;publico;ICMP Destination Unreachable Communication with Destination Host is
    Administratively Prohibited;0
2010-04-12;20:40:59;tcp;-;windows;mail;41.225.140.245;1957;193.136.62.71;135;;TUNISIA;
  ISEP;publico;ICMP Destination Unreachable Communication with Destination Host is
    Administratively Prohibited;0
2010-04-12;20:41:00;tcp;-;linux;web;41.225.140.245;1984;193.136.62.74;135;;TUNISIA;
  ISEP;publico;ICMP Destination Unreachable Communication with Destination Host is
    Administratively Prohibited;0
2010-04-12;20:41:00;tcp;-;linux;web;41.225.140.245;1984;193.136.62.74;135;;TUNISIA;
  ISEP;publico;ICMP Destination Unreachable Communication with Destination Host is
    Administratively Prohibited;0
2010-04-12;20:41:01;tcp;-;linux;web;41.225.140.245;1984;193.136.62.74;135;;TUNISIA;
  ISEP;publico;ICMP Destination Unreachable Communication with Destination Host is
    Administratively Prohibited;0
2010-04-12;22:26:08;tcp;-;windows;mail;41.225.140.245;1620;193.136.62.71;135;;TUNISIA;
  ISEP;publico;ICMP Destination Unreachable Communication with Destination Host is
```

```
    Administratively Prohibited;1
2010-04-12;22:26:08;tcp;-;linux;mail;41.225.140.245;1622;193.136.62.73;135;;TUNISIA;
  ISEP;publico;ICMP Destination Unreachable Communication with Destination Host is
    Administratively Prohibited;0
2010-04-12;22:26:08;tcp;-;windows;web;41.225.140.245;1621;193.136.62.72;135;;TUNISIA;
  ISEP;publico;ICMP Destination Unreachable Communication with Destination Host is
    Administratively Prohibited;0
2010-04-12;22:26:08;tcp;-;linux;web;41.225.140.245;1623;193.136.62.74;135;;TUNISIA;
  ISEP;publico;ICMP Destination Unreachable Communication with Destination Host is
    Administratively Prohibited;0
2010-04-12;22:26:08;tcp;-;linux;mail;41.225.140.245;1622;193.136.62.73;135;;TUNISIA;
  ISEP;publico;ICMP Destination Unreachable Communication with Destination Host is
    Administratively Prohibited;0
2010-04-12;22:26:08;tcp;-;linux;web;41.225.140.245;1623;193.136.62.74;135;;TUNISIA;
  ISEP;publico;ICMP Destination Unreachable Communication with Destination Host is
    Administratively Prohibited;0
2010-04-12;22:26:08;tcp;-;windows;mail;41.225.140.245;1620;193.136.62.71;135;;TUNISIA;
  ISEP;publico;ICMP Destination Unreachable Communication with Destination Host is
    Administratively Prohibited;0
2010-04-12;22:26:08;tcp;-;windows;web;41.225.140.245;1621;193.136.62.72;135;;TUNISIA;
  ISEP;publico;ICMP Destination Unreachable Communication with Destination Host is
    Administratively Prohibited;0
2010-04-12;22:26:09;tcp;-;linux;mail;41.225.140.245;1622;193.136.62.73;135;;TUNISIA;
  ISEP;publico;ICMP Destination Unreachable Communication with Destination Host is
    Administratively Prohibited;0
2010-04-12;22:26:09;tcp;-;linux;web;41.225.140.245;1623;193.136.62.74;135;;TUNISIA;
  ISEP;publico;ICMP Destination Unreachable Communication with Destination Host is
    Administratively Prohibited;0
2010-04-12;22:26:09;tcp;-;windows;web;41.225.140.245;1621;193.136.62.72;135;;TUNISIA;
  ISEP;publico;ICMP Destination Unreachable Communication with Destination Host is
    Administratively Prohibited;0
```